

# Conformational Switching in Self-assembling Mechanical Systems: Theory and Application

by

Kazuhiro Saitou

B. Eng., Mechanical Engineering  
University of Tokyo (1990)

S. M., Mechanical Engineering  
Massachusetts Institute of Technology (1992)

Submitted to the Department of Mechanical Engineering  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

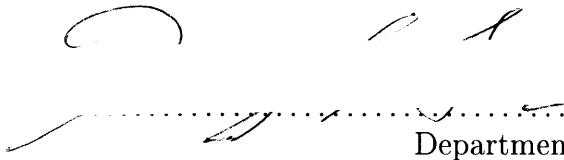
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

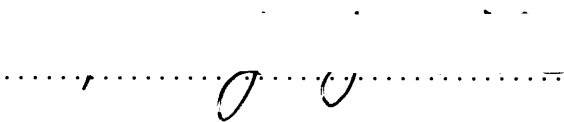
© Massachusetts Institute of Technology 1996. All rights reserved.

Author.....

  
.....  
Department of Mechanical Engineering

May 17, 1996

Certified by .....

  
.....

Mark J. Jakiela  
Associate Professor  
Thesis Supervisor

Accepted by .....

  
.....

Ain A. Sonin  
Chairman, Departmental Graduate Committee

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

JUL 17 1996

LIBRARIES

# Conformational Switching in Self-assembling Mechanical Systems: Theory and Application

by

Kazuhiro Saitou

Submitted to the Department of Mechanical Engineering  
on May 17, 1996, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy in Mechanical Engineering

## Abstract

This work is directed towards the design of *self-assembling* mechanical systems – mechanical systems that can be assembled via random interactions among their components. A class of self-assembling mechanical systems is studied where assembly instructions are written in each component in terms of *conformational switches*. Conformational switches are mechanisms which change component shape as a result of local interactions with other components, observed in the spontaneous self-assembly of bacteriophages.

Parametric design optimization of two types of mechanical conformational switches is discussed. These mechanical conformational switches are used as building blocks of parts for one-dimensional self-assembly via *sequential random bin-picking*. A genetic algorithm, in conjunction with computer simulation of sequential random bin-picking, optimizes the parameterized switch designs maximizing the yield of a desired assembly. The results are presented in the case of two, three and four part one-dimensional self-assembly. Rate equation analyses of the resulting designs reveal that conformational switches can change part concentration by forming temporal intermediate assemblies, and can encode sub-assembly sequences. Effects of initial part concentration and defects during assembly are discussed. Design guidelines for  $n$ -part self-assembling systems are made based on these results, and principle of subassembly in biology is re-examined in the context of self-assembling mechanical systems.

An abstract model of self-assembling systems is presented where assembly instructions are written as local rules that specify conformational changes of components. The model, *self-assembling automaton*, is defined as a sequential rule-based machine that operates on strings of symbols. An algorithm is provided for constructing a self-assembling automaton which self-assembles a string of distinct symbols in a given sequence. Classes of self-assembling automata are defined based on three classes of subassembly sequences described by *assembly grammars*. The minimum number of conformations is provided which is necessary to encode instances of each class of subassembly sequences. It is proven that the rules corresponding to the above two types of conformational switches, and three conformations for each component, can encode subassembly sequences of a string of distinct symbols with arbitrary length.

Finally, an example is presented where the concept of conformational switching is applied to assembly in micro-electromechanical systems (MEMS).

Thesis Supervisor: Mark J. Jakiela  
Title: Associate Professor

## Acknowledgements

This work is by no means a solitary effort. I am indebted to many people for technical and personal support during the research process.

I cannot thank enough my thesis advisor, Professor Mark Jakiela, for giving me an opportunity to work on this *very* exciting project. It is no doubt that his continuous support, guidance, and encouragement made this work possible. Special thanks go to the members of my thesis committee: Professors David Gossard, Jonathan King, and Thomás Lozano-Pérez. Their comments and suggestions were very helpful towards the completion of my thesis. I gratefully acknowledge Professors Stephen Senturia and Marty Schmidt for making their micro fabrication facilities available. I also acknowledge Dr. G. K. Ananthasuresh and Mr. Charles Hsu for the fabrication of the micro mouse trap. I thank Professor Seth Lloyd for valuable comments on the theory part of this work.

This work is supported by the National Science Foundation with a Presidential Young Investigator's grant (DDM-9058415). Matchable funds for this grant have been provided by Schlumberger Inc.. The partial support for the author has been provided by Zexel Corporation. The simulation part of this work was carried out using the computational facilities of the Computer-Aided Design Laboratory at the Massachusetts Institute of Technology. The design and fabrication of the micro mouse trap was performed using the micro fabrication facilities of the Microsystems Technology Laboratory at the Massachusetts Institute of Technology. These sources of support are gratefully acknowledged.

I would like to thank members of MIT CADLAB, past and present, for the precious interaction and experience I received while working there. Special thanks to Jayaraman Krishnasamy and Rahul Dige, my officemates in 3-452 since 1992, who had made my life at MIT a fun experience. I thank Jay for many valuable comments he made at the practice presentations in the previous night of my thesis defense. I also thank Narendra Soman, Francis Pahng and Thomas Grueninger for the useful comments. Special thanks should go to Hunwook Lim, my exercise partner, for "real" Korean dinner after the defense.

I would like thank my family for their love, support, and encouragement throughout my graduate school career at MIT. Oka-san, Oto-san, I promise I am getting a real job soon! Last, and most important, I thank Kyoko for her friendship, support and love. Kyoko, you have contributed to this work in *much* more ways than you will ever know: *I love you.*

*To Kyoko*

# Contents

<b>1</b>	<b>Introduction</b>	<b>15</b>
1.1	Coded and uncoded self-assembly . . . . .	16
1.2	Principle of subassembly . . . . .	17
1.3	Applications of self-assembly in mechanical engineering . . . . .	18
1.3.1	Assembly at very high rates . . . . .	20
1.3.2	Assembly of very small things . . . . .	21
1.3.3	Assembly very far away . . . . .	21
1.4	Scope of this thesis . . . . .	21
1.5	Overview of the Chapters . . . . .	22
<b>2</b>	<b>Related Work</b>	<b>24</b>
2.1	Self-assembling mechanical systems . . . . .	24
2.1.1	Self-positioning of small mechanical parts . . . . .	24
2.1.2	Mechanical model of conformational switches . . . . .	28
2.2	Self-assembling chemical systems . . . . .	30
2.2.1	Liposome and self-assembling monolayers . . . . .	30
2.2.2	Three-dimensional self-assembly of synthetic supermolecules . . . . .	33
2.3	Self-assembling biological systems . . . . .	35
2.3.1	Computational models of bacteriophage assembly . . . . .	35
2.3.2	Models of subassembly processes in biological assembly . . . . .	36
<b>3</b>	<b>Parametric Design Optimization of Simple Mechanical Conformational Switches</b>	<b>39</b>

3.1	Sequential random-bin picking . . . . .	40
3.2	Mechanical conformational switch for one-dimensional self-assembly . . . . .	42
3.2.1	Sliding bar mechanism . . . . .	42
3.2.2	Minus devices . . . . .	44
3.3	Parametric design optimization with Genetic Algorithms . . . . .	50
3.3.1	Genetic Algorithms . . . . .	50
3.3.2	Problem formulation . . . . .	52
3.3.3	Reinforcement evaluation . . . . .	53
3.4	Two part one-dimensional self-assembly with sliding bar mechanisms . . . . .	53
3.4.1	Design parameterization . . . . .	54
3.4.2	Two part sequential assembly . . . . .	54
3.4.3	Two part self-assembly . . . . .	58
3.4.4	Rate equation analyses of two part self-assembly . . . . .	59
3.4.5	Two part self-assembly with a dummy part . . . . .	62
3.4.6	Rate equation analyses of two part self-assembly with a dummy part . . . . .	64
3.5	Subassembly generation in multi-part one-dimensional self-assembly with minus devices . . . . .	67
3.5.1	Design parameterization . . . . .	69
3.5.2	Representaion of subassembly sequences . . . . .	71
3.5.3	Three part one-dimensional self-assembly . . . . .	72
3.5.4	Rate equation analyses of three part self-assembly . . . . .	74
3.5.5	Four part self-assembly . . . . .	83
3.5.6	Rate equation analyses of four part self-assembly . . . . .	89
3.5.7	Sensitivity of yield to the initial concentration of parts . . . . .	97
3.5.8	Encoding power of a conformational switch model . . . . .	99
3.6	Summary . . . . .	103

<b>4</b>	<b>Theory of One-dimensional Self-assembling Automata</b>	<b>105</b>
4.1	Motivation . . . . .	106
4.2	Conformational switches as assembly instructions . . . . .	107
4.3	Definition of one-dimensional self-assembling automata . . . . .	109
4.4	Constructing one-dimensional self-assembling automata . . . . .	113
4.5	Classes of one-dimensional self-assembling automata . . . . .	118
4.6	Minimum conformation self-assembling automata . . . . .	123
4.7	Summary . . . . .	131
<b>5</b>	<b>Conformational Switch for Micro Assembly: Micro “Mouse Trap”</b>	<b>132</b>
5.1	Motivation . . . . .	133
5.2	Mouse trap design . . . . .	133
5.3	Device fabrication . . . . .	137
<b>6</b>	<b>Discussion and Future Work</b>	<b>143</b>
6.1	Summary of the work . . . . .	143
6.2	Discussion . . . . .	145
6.2.1	Bin-picking simulation vs. rate equation analyses . . . . .	145
6.2.2	Limitation of the theory of one-dimensional self-assembling automata	146
6.2.3	Potential applications of micro mouse trap . . . . .	147
6.3	Contributions of this work . . . . .	148
6.4	Future work . . . . .	148
6.4.1	Extensions to the theory of self-assembling automata . . . . .	148
6.4.2	Turing completeness of one-dimensional self-assembling automata . .	149
6.4.3	Three part assembly using APOS . . . . .	149
<b>A</b>	<b>Optimal designs of two part non-randomized assembly</b>	<b>152</b>
<b>B</b>	<b>Rate equation analysis of two part randomized assembly with a dummy</b>	

<b>part</b>	<b>155</b>
<b>C Rosen's subassembly model</b>	<b>163</b>
<b>D Proof of The Unique Factorization Theorem</b>	<b>166</b>
<b>E The size of the space of subassembly sequences</b>	<b>168</b>



## List of Figures

1-1	Biological example of conformational switching . . . . .	17
1-2	Pathway of T4 bacteriophage assembly . . . . .	19
2-1	Sony's Automated Parts Orienting System (APOS). . . . .	25
2-2	Layered palletization technique. . . . .	25
2-3	Fluidic self-assembly of GaAs blocks onto Si substrates. . . . .	26
2-4	Vibratory self-assembly of hexagonal parts. . . . .	27
2-5	Two-dimensional self-assembly of micro parts using surface tension of the water. . . . .	27
2-6	Self-adjusting microstructures (SAMS). . . . .	28
2-7	Snap-lock mechanism of a hinged microstructure. . . . .	29
2-8	Micro snap fastener. . . . .	29
2-9	Self-reprodcng machines. . . . .	31
2-10	Self-assembling triangles. . . . .	32
2-11	Cross-sectional view of liposome. . . . .	32
2-12	Self-assembling monolayer (SAM). . . . .	33
2-13	Cylindrical supermolecuar structure assembled via metal coordination. . . .	34
2-14	Dimerization of self-complementary subunits to form a molecular "tennis ball." 34	
2-15	Tail structure and assembly of bacteriophage T4 . . . . .	35
3-1	Sequential random bin-picking. . . . .	41
3-2	One-dimentional conformational switch with a sliding bar mechanism. . . .	43
3-3	Bond configurations. . . . .	44

3-4	Three types of bonding. . . . .	45
3-5	Propagation of conformational changes and detaching. . . . .	45
3-6	Priority to upstream propagation. . . . .	46
3-7	One-dimensional conformational switch with a minus device. . . . .	48
3-8	One-dimensional conformational switch with “two-digit” bonding sites. . . . .	49
3-9	Examples of bond configurations and the corresponding two-digit bonding site shapes (of the left bonding site). . . . .	49
3-10	Examples of three types of bonding. . . . .	51
3-11	Priority to upstream propagation. . . . .	52
3-12	Bit assignment of a chromosome. . . . .	55
3-13	Deterministic robot bin-picking. . . . .	56
3-14	Best designs for two part, sequential assembly, found by GA. . . . .	57
3-15	Best design (part $A$ : part $B = 1 : 1$ ) : Design I . . . . .	58
3-16	Best design (part $A$ : part $B = 4 : 1$ ) . . . . .	60
3-17	Solution of Equation 3.5 ( $A : B = 1 : 1$ ). . . . .	63
3-18	Yield computed by Equation 3.5 ( $A : B = 1 : 1$ ). . . . .	64
3-19	Solution of Equation 3.5 ( $A : B = 4 : 1$ ). . . . .	65
3-20	Yield computed by Equation 3.5 ( $A : B = 4 : 1$ ). . . . .	66
3-21	Best design ( $A : B : C = 8 : 1 : 1$ ) : Design I. . . . .	66
3-22	Best design ( $A : B : C = 11 : 3 : 11$ ) : Design II. . . . .	67
3-23	Yield computed by Equation 3.5 ( $A : B : C = 8 : 1 : 1$ ). . . . .	68
3-24	Yield computed by Equation 3.5 ( $A : B : C = 11 : 3 : 11$ ). . . . .	68
3-25	Bit assignment of a chromosome. . . . .	70
3-26	Example of parameter encoding of a part. . . . .	70
3-27	Representations of subassembly sequences. . . . .	71
3-28	Best designs with $\mathbf{n}_0 = (10, 10, 10)$ . . . . .	73
3-29	Best designs with $\mathbf{n}_0 = (10, 20, 10)$ . . . . .	75

3-30 Best designs with $\mathbf{n}_0 = (20, 20, 10)$ . . . . .	76
3-31 Solution with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.0, 0.0)$ . . . . .	80
3-32 Solution with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0)$ . . . . .	80
3-33 Solution with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.0, 0.2)$ . . . . .	81
3-34 Solution with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.0, 0.0)$ . . . . .	82
3-35 Solution with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$ . . . . .	82
3-36 Solution with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.0, 0.2)$ . . . . .	83
3-37 Solution with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.0, 0.0)$ . . . . .	84
3-38 Solution with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$ . . . . .	84
3-39 Solution with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2)$ . . . . .	85
3-40 Five non-ambiguous subassembly sequences of a four part assembly. . . . .	86
3-41 Best designs with $\mathbf{n}_0 = (10, 10, 10, 10)$ . . . . .	87
3-42 Best designs with $\mathbf{n}_0 = (10, 20, 10, 10)$ . . . . .	88
3-43 Best designs with $\mathbf{n}_0 = (10, 20, 20, 10)$ . . . . .	90
3-44 Eight possible subassembly sequences. . . . .	92
3-45 Solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$ . . . . .	93
3-46 Solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$ . . . . .	93
3-47 Solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$ . . . . .	94
3-48 Solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.2)$ . . . . .	94
3-49 Solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$ . . . . .	95
3-50 Solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$ . . . . .	95
3-51 Solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$ . . . . .	96
3-52 Solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.2)$ . . . . .	96
3-53 Solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$ . . . . .	97
3-54 Solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$ . . . . .	98
3-55 Solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$ . . . . .	98
3-56 Solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.2)$ . . . . .	99

3-57	Solution with $\mathbf{n}_0 = (10, 11, 11, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$ . . . . .	100
3-58	Solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$ : comparison with un-encodable subassembly sequences . . . . .	101
3-59	Conformational switch design that encodes $\{A(BC)D\}$ . . . . .	101
3-60	Conformational switch design that encodes $((A(BC))D)$ . . . . .	102
3-61	Conformational switch design that encodes $(\{ABC\}D)$ . . . . .	103
4-1	One-dimensional self-assembly with no conformational switches. . . . .	108
4-2	One-dimensional self-assembly with conformational switches . . . . .	109
4-3	Parse tree of an assembly template generated by $G_I$ . . . . .	120
4-4	Parse tree of an assembly template generated by $G_{II}$ . . . . .	124
5-1	Schematic top view of the micro mouse trap. . . . .	134
5-2	The four steps of the action of the micro mouse trap. . . . .	135
5-3	Solid model for the prototype micro mouse trap. . . . .	136
5-4	FEM result of the outer latch of the prototype mouse trap. . . . .	138
5-5	FEM result of the inner latch of the prototype mouse trap. . . . .	138
5-6	FEM result of the inner latch of the prototype mouse trap. . . . .	139
5-7	Fabrication process flow. . . . .	140
5-8	Structure mask layout. . . . .	141
5-9	Prototype micro mouse trap (SEM micrograph). . . . .	142
6-1	Three part self-assembly using APOS. . . . .	150
A-1	Optimal designs for two-part sequential assembly. . . . .	153
A-2	Equivalent designs of part B. . . . .	154
B-1	Solution of equation 3.5 for Design I ( $A : B : C = 8 : 1 : 1$ ). . . . .	159
B-2	Solution of equation 3.5 for Design II ( $A : B : C = 8 : 1 : 1$ ). . . . .	160
B-3	Solution of equation 3.5 for Design I ( $A : B : C = 11 : 3 : 11$ ). . . . .	161

B-4 Solution of equation 3.5 for Design II ( $A : B : C = 11 : 3 : 11$ ). . . . . 162

## List of Tables

3.1	Summary of the results: three part self-assembly. . . . .	74
3.2	Summary of the results: four part self-assembly. . . . .	89

---

# Introduction

---

This work is directed towards the design of *self-assembling* mechanical systems – mechanical systems that can be assembled via random interactions among their components. In particular, this thesis focuses a class of self-assembling systems where assembly instructions are written in each component in terms of *conformational switches*. Conformational switches are mechanisms which cause *conformational switching* (shape changes) of components as a result of interaction with other components. Conformational switching of the component protein molecules, for example, is observed in the spontaneous self-assembly of bacteriophages. The goal of this thesis is the fundamental understanding of the role of such conformational switching in self-assembling systems, and application of conformational switching to design of self-assembling mechanical systems.

This chapter will first introduce the terms *coded* and *uncoded* self-assembly, and the principle of subassembly, an important characteristic of coded self-assembly. The potential applications of self-assembling mechanical systems will be then described. Also, the scope of this thesis will be explained. This chapter concludes with a short description of the remaining chapters in the thesis.

## 1.1 Coded and uncoded self-assembly

In [59], Whitesides provided a definition of a self-assembling system:

A self-assembling process is one in which humans are *not* actively involved, in which atoms, molecules, aggregates of molecules and components arrange themselves into ordered, functioning entities without human intervention.

Nature exhibits various kinds of self-assembly. One of the simplest is raindrops on a leaf which, when placed close enough, merge together spontaneously to form one big drop with smooth, curved shape. On the other extreme in complexity, protein molecules inside biological cells self-assemble to reproduce cells each time they divide. These two examples represent two types of self-assembly in nature – *coded* and *uncoded* self-assembly [59]. The self-assembly of raindrops is an example of *uncoded* self-assembly, where assembly of each component (in this case each raindrop) is directed simply by minimization of potential (in this case thermodynamic) energy. Uncoded self-assembly, therefore, works to construct only the simplest of such structures. On the other hand, many complex structures in nature, *e.g.* biological cells, arise via *coded* self-assembly, where instructions for the assembly of the system are built into its components. Self-assembly of biological cells, for example, is directed by conformational changes in protein molecules realized by energy-dissipating structures such as ATP.

A well-studied example of coded self-assembly in biology is assembly of bacteriophages, a type of virus which infect bacterial cells. It is known that the assembly of new progeny viruses in their host cell occurs in a fixed morphogenetic pattern, indicating coded self-assembly. Biologists believe that assembly instructions for this self-assembly of bacteriophages are written in each component molecules in the form of *conformational switches*. In a protein molecule with several bond sites, 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 which occurred at an assembly step provides the essential substrate for assembly at the next step [58].

Figure 1-1 shows an example of a conformational switch in the inhibition of protein enzyme by specific end-product inhibitor molecules. The catalytic activity of many protein enzyme are activated by binding their substrates, and forming enzyme-substrate complexes



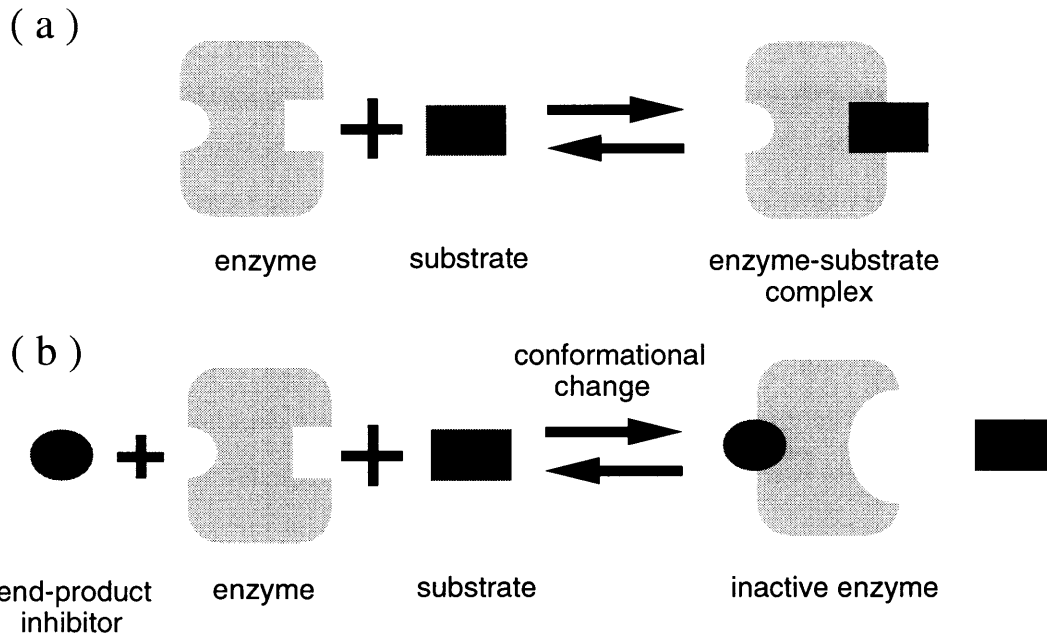


Figure 1-1: Biological example of conformational switching: inhibition of protein enzyme by specific end-product inhibitor molecules (abstracted from Figure 4-21 of [58])

(Figure 1-1-a). The end-product inhibitor blocks the enzyme activity by reversibly binding to the enzyme at a site other than the active site (the region that binds the substrates). The binding of the inhibitor causes a conformational change at the active site on the enzyme, which prevents the enzyme from combining with its substrate (Figure 1-1-b). The chemical forces binding a specific end-product inhibitor to an enzyme are weak secondary forces such as hydrogen bonds, salt linkages, and van der Waals forces and do not involve covalent bonding. Hence, inhibition can be quickly reversed once the end-product concentration is reduced to a low level.

## 1.2 Principle of subassembly

Assembly of many complex systems – either biological or mechanical – takes place in several stages. At each assembly stage, *subassemblies* are made, which are then incorporated into subassemblies at the next stage. For example, the beginning stage of an automobile assembly is the production of elementary parts, such as wire, bolts, nuts, etc. These are put together into subassemblies such as generators, dashboard instruments, etc., which are then used

to build more complicated subassemblies such as engines, bodies, etc. This process of subassembly has three significant advantages [19, 18]: *reliability*, *efficiency* and *variety*. Subassembly processes are reliable, since elimination of defective subassemblies can be done at each assembly stage. A defective subassembly produced at one stage will not be built into an assembly at the next stage. Also, with regard to efficiency, subassemblies at a stage can be carried out simultaneously. This greatly speeds up an entire assembly process, compared to sequential step-by-step assembly processes where parts are put together only one at a time. Finally, a common subassembly can be used in a variety of different assemblies, or can be used repeatedly in many places in an assembly. It is clear that these advantages of subassembly process is more significant in self-assembling systems, where assembly occurs via random interactions among components. This seems to be the reason almost all coded self-assembling systems in nature show the formation of subassemblies.

An example subassembly process in biological systems is found in the self-assembly of bacteriophages. Figure 1-2 shows the simplified pathway for assembly of a type of bacteriophage, T4, which infects the bacterium *E.coli* strain B [9, 14, 10]. The morphogenetic pathway clearly shows formation of subassemblies; a head and a tail form a head-tail subassembly, and this subassembly is then put together with tail fibers. For simplicity, let us write the assembly tree in Figure 1-2 in a list representation ( $F(TH)$ ), where  $F$ ,  $T$  and  $H$  represent tail fiber, tail and head, respectively. A question arises immediately: how did nature prefer the fixed subassembly sequence ( $F(TH)$ ) out of all possible subassembly sequences? In particular, why not the other possible sequence,  $((FT)H)$ ? What kind of conformational switching realizes this fixed morphogenetic pattern?

### 1.3 Applications of self-assembly in mechanical engineering

Based on the discussion in the preceding sections, it can be said that the process of self-assembly holds the following characteristics:

- **Random interaction among components:** assembly occurs via the random interaction among components, subassemblies, and possibly final assemblies.
- **Asynchronized formation of parallel assembly:** multiple subassemblies can form parallelly and the formation of each subassembly proceeds independently.

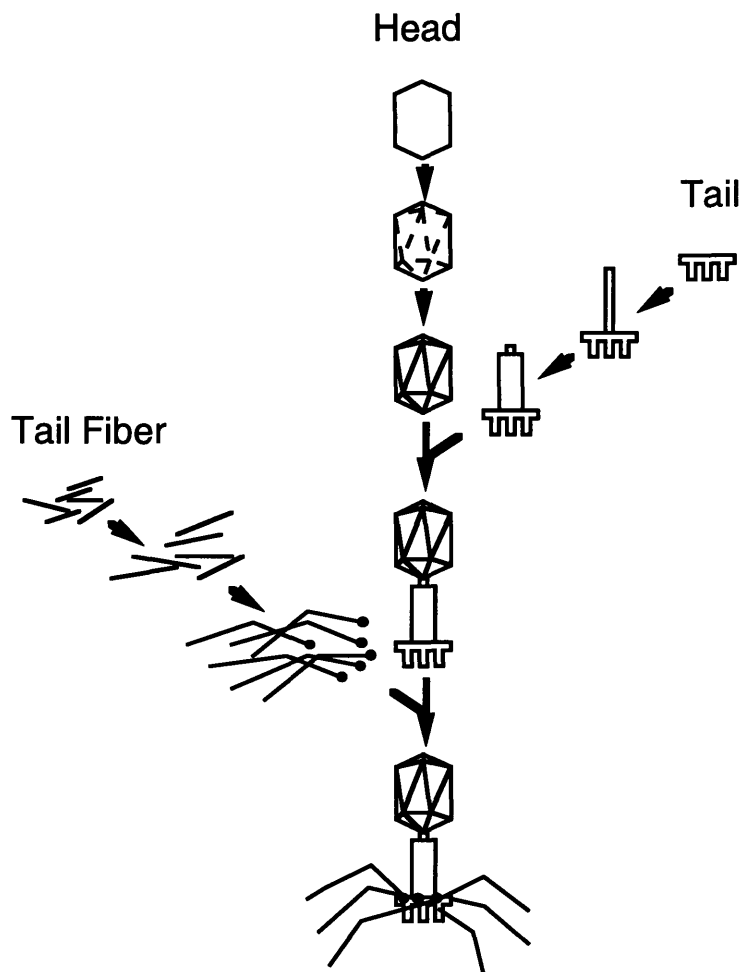


Figure 1-2: Pathway of T4 bacteriophage assembly (simplified and rearranged from Figure 3 of [54])

From mechanical engineering point of view, these characteristics highlight a number of attractive features of self-assembly, in particular coded self-assembly. These include:

- **Efficiency:** since the process of self-assembly implicitly realizes parallel assembly of components, very high assembly rates can be achieved. Also, no need for precise part positioning/orientation increases the efficiency of assembly process, especially in assembly of very small parts.
- **Robustness:** since assembly instructions are distributed among each components, the assembly is more robust against unknown disturbances to the systems. This makes the systems suitable to automated assembly in unstructured/unmodeled environment.

The following discusses some areas of the practical applications of coded self-assembly in mechanical engineering which utilize the above features.

### 1.3.1 Assembly at very high rates

The common image of assembly is a robotic or human hand grasping one part at a time, assembling it into a product held on a fixture. There are natural limits to the speed with which this process can be carried out. Adding more human or robot hands is often not cost effective to achieve very high rates of assembly.

Moncevicz *et al.* [38, 40, 39] developed a layered palletization technique, where parts are “palletized” by using vibration to convey them over a plastic “pallet” into which are carved an array of relief shapes that trap and orient the flowing parts. The first part is designed such that once a quantity is held in the pallet, it becomes integral with the pallet for the purpose of palletizing a quantity of the second part. Therefore, the second part palletization actually assembles the second part to the first part. Since many part insertions occur simultaneously, a very high assembly rate can be achieved. I believe that this layered palletization process (and other similar processes) could be employed in more cases if temporary nonfunctional shape features could be added to the pallet or the palletized parts. Between part layers, for example, a “primer layer” of nonfunctional “pseudo parts” could be palletized. The sole purpose of this primer layer would be to cause geometric shapes that would facilitate the subsequent palletization of the next part layer. If the pseudo parts are not needed

for (or worse prevented) the functioning of the assembled device, they must be removed before the device is used. This could be accomplished if their design involved the use of a conformational switch that detached them upon the arrival and assembly of the part whose assembly they were intended to facilitate.

### **1.3.2 Assembly of very small things**

Another problem with the common “part grasping” image of assembly is that some parts are simply too small to grasp. In that case, a self-assembly process might be the only possible approach. Yeh and Smith [63, 64] used a (non-layered) palletization technique similar to [40, 39], to assemble microstructures. They fabricated trapezoidal gallium arsenide (GaAs) blocks and a Si wafer with trapezoidal holes. Assembly is then done by releasing the GaAs blocks in a carrier fluid (ethanol) and dispensing the fluid over the Si wafer. Cohn *et al.* [11] experimented with the self-assembly of a small hexagonal lattice (1 mm in diameter) by placing a quantity of them on a slightly concave diaphragm that was agitated with a loudspeaker. Incorporating conformational switching to such micro-scale self-assembly processes might facilitate the non-trivial assembly of very small parts.

### **1.3.3 Assembly very far away**

Consider trying to remotely assemble a set of components that are very far away. An example might be the construction of housing units on a distant planet. Teleoperation control issues arise along with the problem of the cost of transmitting control signals back and forth to the remote site. An alternate approach might be to send quantities of the component parts to the site, and outfit them with an autonomous means of locomotion. The parts could roam around until they find each other and correctly assemble. Again, the similarities with the viral assembly process are clear, and conformational switches could be similarly useful.

## **1.4 Scope of this thesis**

The goal of this thesis is the fundamental understanding of the role of conformational switching in self-assembling systems, and application of conformational switching to design

of self-assembling mechanical systems. More specifically, the following problems will be addressed:

- What is the role of conformational switching in the self-assembling systems, where assembly occurs via random interaction among components? What kinds of conformational switches are necessary to facilitate complex self-assembling processes?
- What is the mechanical implementations of conformational switches like? How one can optimize the design of self-assembling mechanical systems using the mechanical conformational switches?

Due to the random interactions among components, the process of self-assembly can be very difficult to model. The modeling of bacteriophage self-assembly, for example, involves modeling of dynamic interaction of molecules driven by electrostatic forces and van der Waals forces. Studying such complex self-assembling processes as it is, therefore, is not feasible for the purpose of addressing the above problems. Alternative approach is to define a simplified model of self-assembling processes and use it as a tool to study the above problems. If the simplified model effectively captures the essence of self-assembling process in general, *i.e.* random interaction among components and asynchronized formation of parallel assembly, what is learned from studying the simplified model will be able to give the useful insight in the case of more complex self-assembling systems. Accordingly, this thesis will address the above problems in the case of a simple model of self-assembling processes called *sequential random bin-picking*. It is a one-dimensional self-assembling process where assembly of a random pair of components occurs at a time. The details of the model will be explained in Section 3.1.

## 1.5 Overview of the Chapters

The next chapter describes the previous work on self-assembling systems in mechanical engineering, chemistry, and biology. In mechanical engineering, several designs of mechanical systems with self-positioning/self-orientation capability are discussed. Also, some work is presented on the mechanical model of conformational switches. In chemistry, work on deigning three-dimensional self-assembling structure via molecular aggregation is described.

In biology, work on computational model of bacteriophage assembly, and classical models of subassembly processes in biological self-assembly is discussed.

Chapter 3 discusses parametric design optimization of two types of mechanical conformational switches: sliding bar mechanisms and minus devices. It first introduces the sequential random bin-picking, the simplified model of self-assembling process used throughout this thesis. Then it briefly explains the genetic algorithm which is used for parametric design optimization. Then, the results of genetic design optimization, along with rate equation analyses of the resulting designs, are presented in the case of two, three and four part one-dimensional self-assembly. Some questions on relationship between conformational switch design and subassembly sequences are discussed.

Chapter 4 describes the theory of one-dimensional self-assembling automata, an abstracted model of self-assembling systems motivated by the questions posed in Chapter 3. It starts with definition of self-assembling automata. An algorithms to construct a self-assembling automata is then presented. Several theorems are proved on the classes of self-assembling automata and the self-assembling automata with minimum conformations.

Chapter 5 presents a design of a micro “mouse trap,” a conformational switch for micro assembly. The basic function of the device, the results from structural analysis, and the fabrication process are described.

Chapter 6 summarizes the contribution of the thesis and suggests several future work.

There are five appendices. Appendix A lists all optimal designs of two-part sequential assembly described in Section 3.4.2. Appendix B is on details of the rate equation for two part self-assembly with a dummy part and its numerical results, which appears in Section 3.4.6. Appendix C explains the detailed derivation of Rosen’s subassembly model described in Section 2.3.2. Appendix D provides the proof of the Unique Factorization Theorem used in Section 2.3.2. Appendix E describes the proof of the fact that the size of non-ambiguous subassembly sequences of n-part one-dimensional assembly is  $\Omega(2^n)$ .

---

## Related Work

---

Although no work has been found directly focusing the scope of this thesis, there are some related work in the general area of self-assembling systems which motivated this work. This section reviews these previous studies on self-assembly in mechanical, chemical and biological systems.

### 2.1 Self-assembling mechanical systems

#### 2.1.1 Self-positioning of small mechanical parts

Several approaches have been proposed to incorporate self-positioning to assembly of small mechanical parts. Some techniques are developed for self-positioning of components from totally random positions and orientations.

Moncevicz and Jakiela developed the concept of mass aggregate assembly [40, 39] in the domain of high volume assembly of consumer electromechanical products (cameras, videocassette recorders, *etc.*). This process employs part presentation devices (feeding and orienting machines) that transport bulk quantities of parts to actually assemble large quantities of parts in parallel simultaneously.

As an implementation of the concept of mass aggregate assembly, Moncevicz [38] de-



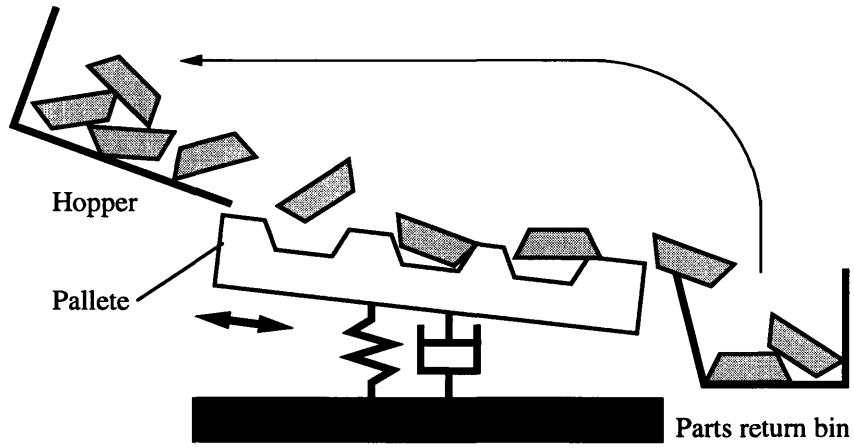


Figure 2-1: Sony's Automated Parts Orienting System (APOS).

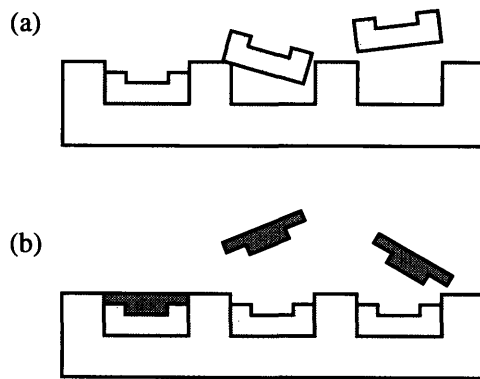


Figure 2-2: Layered palletization technique.

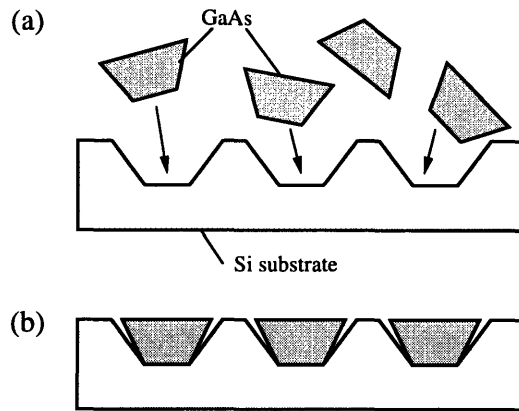


Figure 2-3: Fluidic self-assembly of GaAs blocks onto Si substrates. (redrawn from [64])

veloped layered palletization technique using SONY's Automated Parts Orienting System (APOS). As shown in Figure 2-1, APOS "palletizes" parts by using vibration to convey them over a plastic pallet with an array of relief shapes that trap and orient the flowing parts. To achieve parallel assembly of two parts, a quantity of the first part is palletized (Figure 2-2(a)). This first part is designed such that once held in the pallet relief forms, it becomes integral with the pallet for the purpose of palletizing a quantity of the second part. The second part palletization actually assembles a quantity of the second part to the first part (Figure 2-2(b)). The mated part pair is removed as a subassembly unit from the pallet. Since many part insertions occur simultaneously, a very high assembly rate can be achieved.

In the domain of micro-electromechanical systems (MEMS), Yeh and Smith [63, 64] assembled light-emitting diodes (LED) using a (non-layered) palletization technique similar to [40, 39]. Trapezoidal microstructures made of gallium arsenite (GaAs) are transferred into an inert carrier fluid (ethanol) and assembled onto a host silicon (Si) substrate with trapezoidal holes by fluid transport (see Figure 2-3). This technique is well suited for the integration of microstructures on substrates made of incompatible material systems such as GaAs and Si.

Cohn *et al.* [11] experimented with the self-assembly of small hexagonal parts (1 mm in diameter) by placing a quantity of them on a slightly concave diaphragm that was agitated

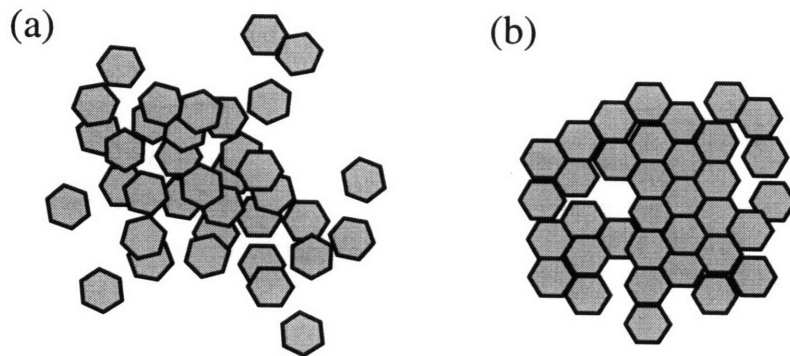


Figure 2-4: Vibratory self-assembly of hexagonal parts. (redrawn from [11])

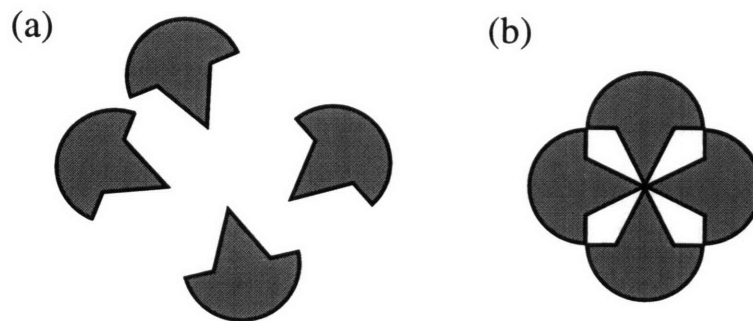


Figure 2-5: Two-dimensional self-assembly of micro parts using surface tension of the water. (redrawn from [29])

with a loudspeaker. The curvature of the diaphragm, as well as the amplitude and frequency of the vibration exerted by the loudspeaker, is controlled so that a pile of the hexagonal parts (Figure 2-4(a)) can form a regular two-dimensional lattice (Figure 2-4(b)) in the process analogous to crystal annealing.

Hosokawa, Shimoyama and Miura [29] experimented with the two-dimensional self-assembly of micro parts on a water surface. The micro parts (approximately  $400\mu m$  long) are designed such that they self-assemble in a regular arrangement due to surface tension of the water (Figure 2-5).

While no external positioning/handling of components is necessary in these techniques, the components are not positively fastened after self-positioning, causing them to disas-

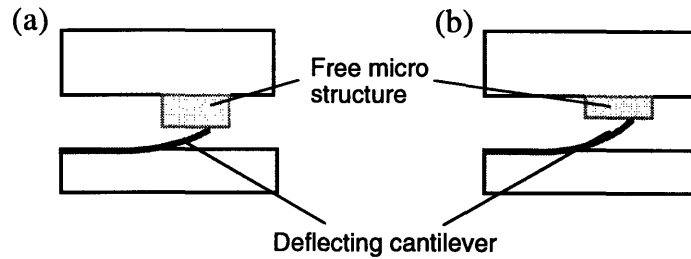


Figure 2-6: Self-adjusting microstructures (SAMS). (redrawn from [32])

semble very easily. On the other hand, some approaches have been proposed on the use of mechanical force to both self-position and fasten micro components so that assembly requires *less* positioning/handling of components. Judy, Cho, Howe and Pisano [32] fabricated a self-adjusting microstructures (SAMS). It is a laterally-deflecting cantilever on the sidewall of a polysilicon mesa which adjusts the position of substrate-free micro structures attaching to the cantilever, and provides the bearing forces between structures (Figure 2-6). Burgett, Pister and Fearing [8] used spring loaded latches to self-position the plates within microfabricated hinges. After fabrication, they used hydrodynamic forces of rinsing water to self-assemble the micro plate to rotate out of the plane of the wafer (Figure 2-7). Prasad, Böhringer and MacDonald [45] fabricated a micro snap fastener with 1–2  $\mu\text{m}$  wide laterally-deflecting chambered latches (Figure 2-8).

### 2.1.2 Mechanical model of conformational switches

Due to its inherent complexity, little work has been done on coded self-assembly of physical systems. Penrose [42], 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 sub-devices 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 (Figure 2-9). Another example is found in Hosokawa *et al.* [28, 27]. 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

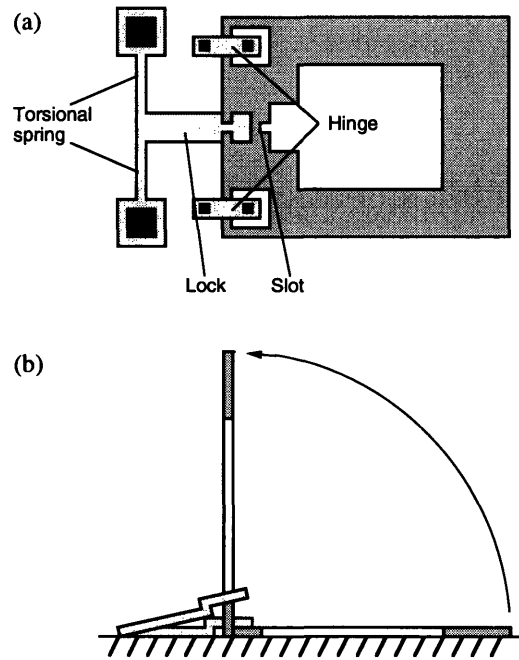


Figure 2-7: Snap-lock mechanism of a hinged microstructure. (redrawn from [8])

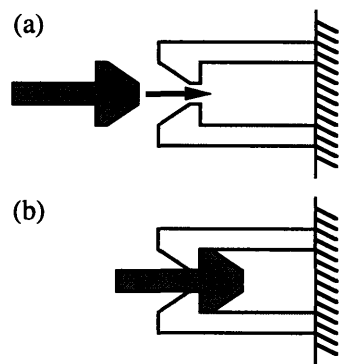


Figure 2-8: Micro snap fastener. (redrawn from [45])

to an inactivated part, turning the part to an activated state (Figure 2-10). 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.

## **2.2 Self-assembling chemical systems**

### **2.2.1 Liposome and self-assembling monolayers**

Since 1960's, biomedical researchers have been experimenting with liposomes, spherical bilayer structure shown in Figure 2-11, as drug-delivery systems. As the name indicates, liposomes are made of phospholipids, major component molecules found in cell membranes. When placed in aqueous environment, phospholipids spontaneously self-assembles a double layer (called phospholipid bilayer), in which the hydrophilic ends are in contact with the water and the hydrophobic ends point towards one another. If there are enough molecules, the phospholipid bilayer will grow into a sphere with a cavity large enough to hold a drug molecule. Since the encapsulated drugs are protected from degradation by enzymes, a drug contained in a liposome envelope can remain active longer than it would otherwise. Liposome drug-delivery systems are currently under clinical trials.

Self-assembling monolayers (SAMs) are designed to utilize the process of uncoded self-assembly similar to liposomes. A SAM is one- to two-nanometer-thick film of synthetic organic molecules that form a two-dimensional crystal on an absorbing substrate. The molecules in a SAM are designed such that they interacts strongly with the surface at one end and not the other, just like the hydrophilic and hydrophobic ends of a phospholipid. The most extensively studied system of SAMs is made of alkanethiols, long hydrocarbon chains with a sulfur atom at one terminus. When a glass plate coated with a thin film of gold is dipped into a solution of alkanethiol, the sulfur atoms attach to the gold, generating a two-dimensional crystal (Figure 2-12). The thickness of this crystal can be controlled by varying the length of the hydrocarbon chain, and the properties of the crystal's surface can be modified by attaching different terminal groups. In contrast to most procedures for surface modification, all these operations are simple and inexpensive, requiring neither high-vacuum equipment nor lithography.

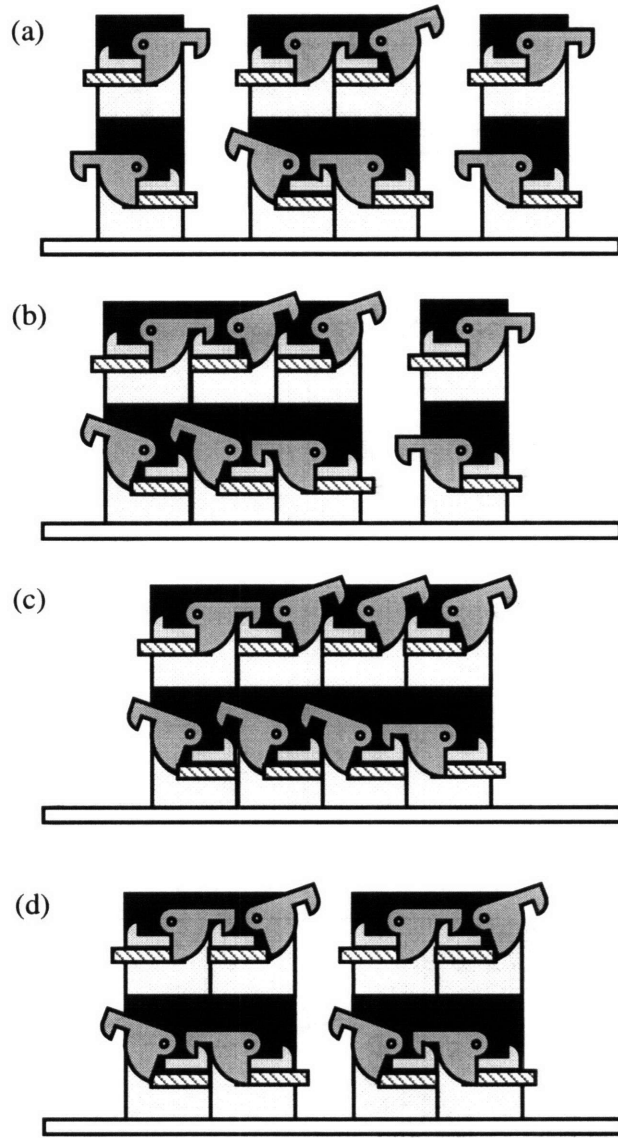


Figure 2-9: Self-reproducing machines. (redrawn from [42])

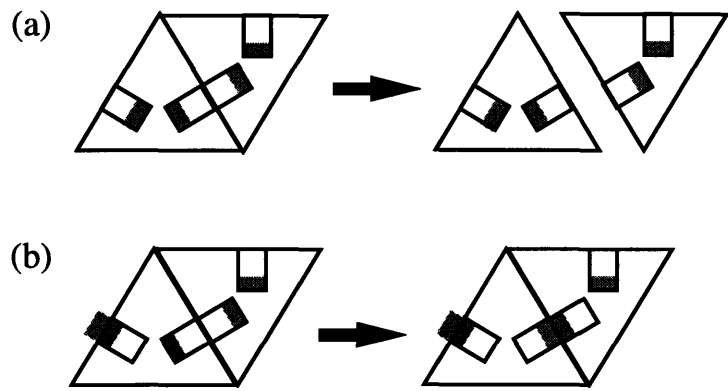


Figure 2-10: Self-assembling triangles. (redrawn from [27])

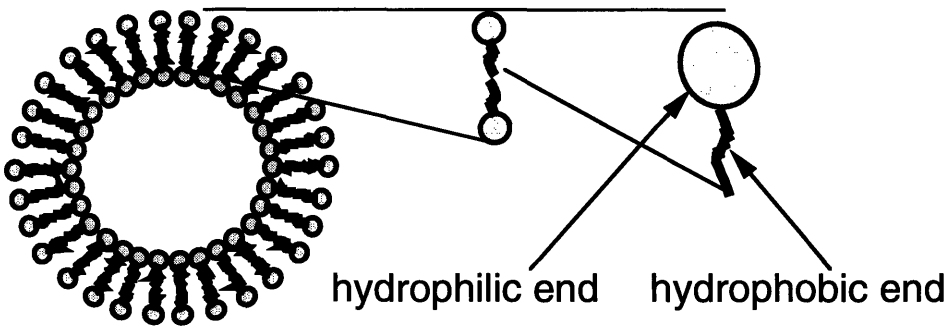


Figure 2-11: Cross-sectional view of liposome. (redrawn from [58])



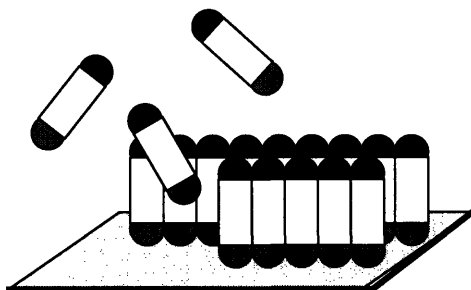


Figure 2-12: Self-assembling monolayer (SAM). (redrawn from [59])

### 2.2.2 Three-dimensional self-assembly of synthetic supermolecules

Uncoded self-assembly of three-dimensional synthetic supermolecules has recently been implemented in several types of organic and inorganic systems. By using metal coordination, hydrogen bonding, and donor-receptor interactions, researchers have achieved the spontaneous formation of three-dimensional supermolecules. Lehn *et.al* introduced several three-dimensional structures assembled via metal coordination. For instance, the cylindrical structure shown in Figure 2-13 is made of three linear structures and two cyclic structures, which are linked by six copper (I) metal ions. The complex form spontaneously upon addition of  $[Cu^I(CH_3CN)_4]BF_4$  to a solution of the organic ligands.

Branda *et. al* [7] designed two complementary molecules which are capable to assembling into dimeric capsules through hydrogen bonding. The capsule is made of two monomeric structures with a self-complementary arrangement of electron donors and receptors on the edge (Figer 2-14(a)). The resulting structure has the same geometry as a tennis ball, as shown in Figure 2-14(b). The self-assembly of the monomeric structures is induced by addition of small guest molecules (*e.g.* xenon), and the resulting dimer encapsulates exactly one guest molecule. The dimerization can be controlled by changes in the acidity of the medium, realizing the reversible encapsulation of the guest molecule.

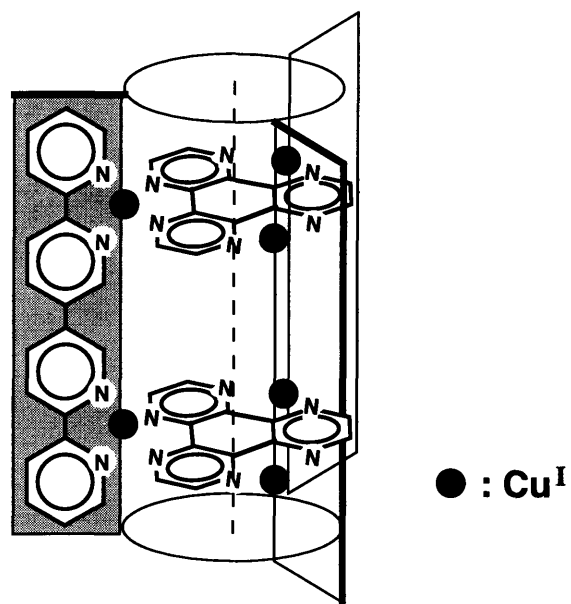


Figure 2-13: Cylindrical supermolecular structure assembled via metal coordination. (redrawn from [34])

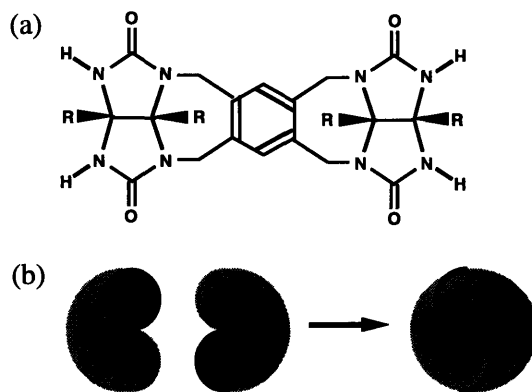
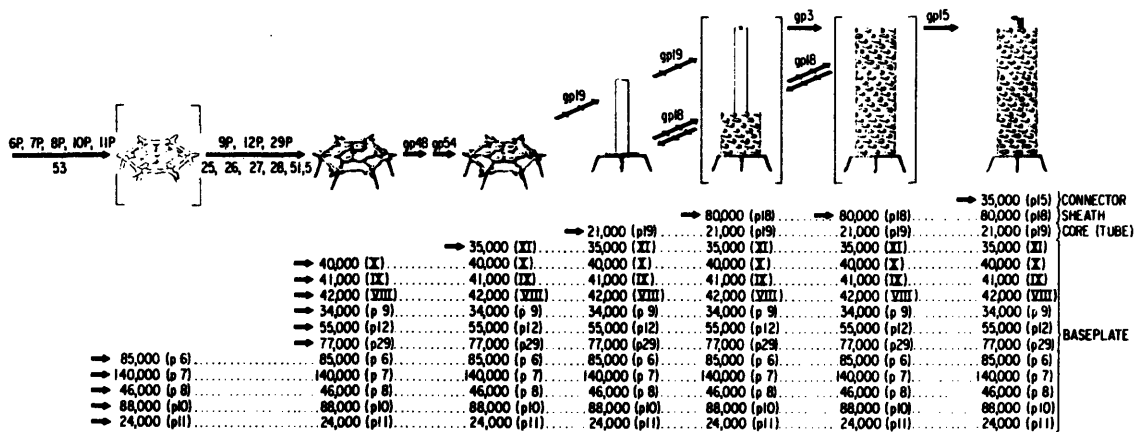


Figure 2-14: Dimerization of self-complementary subunits to form a molecular “tennis ball.” (redrawn from [7])



T4 TAIL STRUCTURE AND ASSEMBLY

Figure 2-15: Tail structure and assembly of bacteriophage T4. (adopted from [9]: see [9, 14] for detail.)

## 2.3 Self-assembling biological systems

### 2.3.1 Computational models of bacteriophage assembly

Coded self-assembly of bacteriophages has been studied by number of biologists (*e.g.* [9, 14], also see Figure 2-15), and several computational models have been developed. Thompson and Goel [54, 55, 19] developed a computer model that simulates the assembly and operation of bacteriophage T4. A simplified model of a virus is constructed from building blocks which are abstractions of the protein molecules. These building blocks are augmented finite automata that can move randomly in their environment and bond to the other blocks. State transition rules of a block specify how bonds can form and how conformational changes propagate within the block. The same approach has been used to model protein biosynthesis [20, 19]. Berger *et al.* [6] developed a local rule theory for self-assembly of icosahedral virus shells. They assume that identical protein subunits take on a small number of distinct conformations. The local rules then specify, for each conformation, which other conformations it can bind to and the approximate interaction angles, interaction length, and torsional angles that can occur between them. By following this local information, the subunits form a closed icosahedral shell with the desired symmetry.

### 2.3.2 Models of subassembly processes in biological assembly

This section discusses two important investigations on subassembly processes in biological assembly. Due to the relevance to this work, these are described in some detail.

#### Crane's subassembly model

Crane [13] has provided a lucid discussion of the advantages of subassembly processes in the construction of complicated structures from elementary subunits. One aspect of his discussion has to do with scheduling the assembly process into a series of stages, or subassembly processes. In his presentation, we start with 1,000,000 identical elementary units. Our objective is to build structures of 1,000 unit length (deca-deca-decamers) using two different subassembly processes. The first subassembly process consists of three stages:

1. Ten elementary units are put together to form 100,000 subassemblies of 10 unit length, or 100,000 decamers.
2. Ten of the decamers are put together to form 10,000 subassemblies of 100 unit length, or 10,000 deca-decamers.
3. Ten of the deca-decamers are put together to form 1,000 final assemblies of 1,000 units long, or 1,000 deca-deca-decamers.

The second subassembly process is to join one thousand elementary units together in a stage. He assumed there was a defect probability that a unit was added wrongly causing the subassembly to become defective. He further assumed the defective subassemblies could not be incorporated into a subassembly at the next stage, and the elementary units in the defective subassemblies were completely wasted. Assuming the same defect probability of 1% at each subassembly stage, the first three-stage subassembly process gives 739 deca-deca-decamers<sup>1</sup> out of a possible 1,000. On the other hand, the second subassembly process with the same defect probability produces only 0.0432 deca-deca-decamer<sup>2</sup>. It is clear that the first three-stage subassembly process is far more efficient than the second one-stage subassembly process.

---

<sup>1</sup>Obtained by  $(1 - 0.01)^{10+10+10}(1,000,000/1,000) = 739$ ; see Appendix C for more detail.

<sup>2</sup>Similarly, by  $(1 - 0.01)^{1,000}(1,000,000/1,000) = 0.0432$ ; see Appendix C for more detail.

## Rosen's subassembly model

Following Crane's work, Rosen [47] posed the following question: how can we choose the number of subassembly stages, and the number of elementary units to be put together at each stage, in such a way as to maximize the yield of the desired assembly at the last stage? By extending Crane's subassembly model, he showed the above problem could be formulated as an integer programming problem<sup>3</sup>:

$$\text{maximize } (1 - q)^{r_1 + r_2 + \dots + r_N} \left( \frac{M}{L} \right) \quad (2.1)$$

$$\text{subject to } L = r_1 \cdot r_2 \cdot \dots \cdot r_N$$

$$N \geq 0; \quad N \in \mathbf{Z} \quad (2.2)$$

$$r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \dots, N\}$$

where  $q$ ,  $M$  and  $L$  are defect probability, the number of elementary units in the initial pool, and the number of elementary units in a desired assembly, respectively.  $N$  is the number of subassembly stages and  $r_i$  is the number of subassemblies produced at the  $(i - 1)$ -th stage, which are incorporated into a subassembly at the  $i$ -th stage<sup>4</sup>. Since  $q$ ,  $M$  and  $L$  are positive constants, and  $(1 - q) \leq 1$ , The above problem is equivalent to minimizing the exponent  $r_1 + r_2 + \dots + r_N$  under the same constraints. The Unique Factorization Theorem<sup>5</sup> in elementary number theory shows that the factorization of  $L$  into prime numbers has the property that sum of the factors is minimal. In the case of  $L = 1000$ , the prime factor decomposition is:

$$L = 1000 = 2 \cdot 2 \cdot 2 \cdot 5 \cdot 5 \cdot 5 \quad (2.3)$$

which gives  $N = 6$ . The corresponding yield is:

$$(1 - 0.01)^{2+2+2+5+5+5} \left( \frac{1,000,000}{1,000} \right) = 810 \quad (2.4)$$

---

<sup>3</sup>The derivation is found in Appendix C

<sup>4</sup>In the Crane's first subassembly process, for example,  $q = 0.01$ ,  $M = 1,000,000$ ,  $L = 1,000$ ,  $N = 3$ , and  $r_1 = r_2 = r_3 = 10$ .

<sup>5</sup>The proof is found in Appendix D.

which is larger than 739, the yield by Crane's first subassembly process. Note that the theorem says nothing about the order of factorization; it makes no difference how we distribute the  $r_i$  among the subassembly stages. In this case, for example,  $(r_1, r_2, r_3, r_4, r_5, r_6) = (2, 2, 2, 5, 5, 5)$  and  $(r_1, r_2, r_3, r_4, r_5, r_6) = (2, 5, 2, 5, 2, 5)$  are equivalent in terms of the yield of the final assemblies.

The importance of Rosen's work lies in the fact that he formulated the problem of finding the best subassembly schedule as an optimization problem, and provided a solution in closed form. There are, however, several points to be generalized to make his model more realistic. First, the defect probability  $q$  should not be the same for each subassembly stage. This generalization, however, yields another integer programming problem, which in general cannot be solved in a closed form:

$$\text{maximize } (1 - q_1)^{r_1} (1 - q_2)^{r_2} \dots (1 - q_N)^{r_N} \left( \frac{M}{L} \right) \quad (2.5)$$

$$\text{subject to } L = r_1 \cdot r_2 \cdot \dots \cdot r_N$$

$$N \geq 0; \quad N \in \mathbf{Z} \quad (2.6)$$

$$r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \dots, N\}$$

The second point for generalization is that the model should allow subassembly processes which involve subassemblies with non-equal numbers of subunits. For instance, it should be possible to consider the subassembly process which produces a 1,000-mer out of five 100-mers and four 125-mers at the final stage. Finally, the model should be extended such that it can express the simultaneous construction of subassemblies. It should, for example, be possible to form a 100-mer out of ten 10-mers *as soon as* the first ten 10-mers are produced, *not after* all the elementary units are assembled into 10-mers.

# Parametric Design Optimization of Simple Mechanical Conformational Switches

---

This chapter discusses parametric design optimization of two types of simple mechanical conformational switches, sliding bar mechanisms and minus devices. These mechanical conformational switches are used as building blocks of self-assembling parts for one-dimensional self-assembly via *sequential random bin-picking*. A genetic algorithm, in conjunction with computer simulation of sequential random bin-picking, optimizes the parameterized switch designs to maximize the yield of a desired assembly. The results of genetic optimization, along with rate equation analyses of the resulting designs, are presented in the case of two, three and four part one-dimensional self-assembly. The principle of subassembly in biology is re-examined in the context of self-assembling mechanical systems, and some questions on relationship between conformational switch design and subassembly sequences are discussed.

### 3.1 Sequential random-bin picking

In this thesis, a simplified model of self-assembling process, the *sequential random bin-picking*, is used to study the role of conformational switching in self-assembling systems. It is a simple one-dimensional self-assembly process where assembly of a randomly-chosen pair of parts (possibly subassembly of parts) occurs at a given time:

Assume a random assortment of parts in a (one-dimensional) bin (Figure 3-1(a)).

**Step 1:** Imaginary robot arm #1 *randomly* picks up a part from the bin. Then, imaginary robot arm #2 randomly picks up another part from the bin (Figure 3-1(b)).

**Step 2:** The two parts are pushed against each other, possibly causing formation of a bond (Figure 3-1(c)).

**Step 3:** The parts are *randomly* returned to the bin (Figure 3-1(d)), possibly as an assembly.

The steps 1-3 are repeated until pre-specified conditions are satisfied (*e.g.* repeat for a specified number of iterations, repeat until the number of parts decreases below a limit, *etc.*). It is assumed that the parts do not change their orientations, so in general,  $AB$  and  $BA$  are two distinct assemblies.

I follow Crane's assumption of defects [13] during the above self-assembly process. Namely, I assume 1) with a certain probability two subassemblies (or two parts) are put together incorrectly, causing the resulting subassembly to be defective, and 2) the defective subassemblies cannot be incorporated into the subsequent subassemblies, so the parts in the defective subassemblies are completely wasted. In the scenario above, a defect could occur at step 2 with a certain probability if two parts picked can form a bond<sup>1</sup>, with the defective subassembly being returned to the bin. If the defective subassembly is chosen at step 1 in subsequent iterations, no bond is formed at the step 2, *regardless of* the bond configurations of the two mating sites. In this manner defective subassemblies "waste" an iteration of the robot bin-picking.

Note that the above definition of the sequential random bin-picking holds the essential

---

<sup>1</sup>If they cannot form a bond, no defect occurs and they are simply returned to the bin.



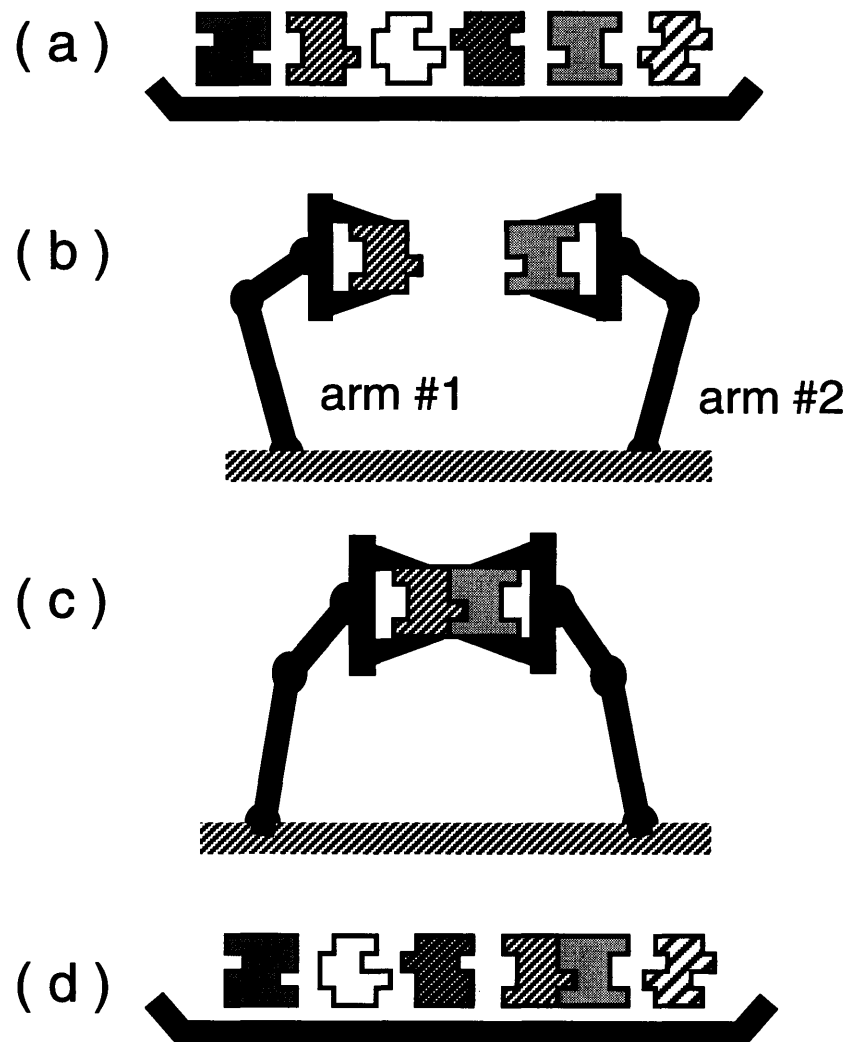


Figure 3-1: Sequential random bin-picking.

characteristics of general self-assembling processes discusses in Section 1.4 – random interactions among components and asynchronized formation of parallel assembly. The part interactions are random due to the random picking at the step 1 and random returning at the step 3<sup>2</sup>. Also, it sequentially simulates asynchronized formation of parallel assembly since the assembly process is “fine-grained” such that only a pairwise mating of two parts, a most fundamental unit of assembly, are allowed at a given time.

Although the sequential random bin-picking is simple enough to be simulated on computer easily, it is also general enough to realize the three generalizations to Rosen’s subassembly model described in Section 2.3.2. It is possible to specify different defect probabilities for different combinations of parts or bond configurations. Also, since the scenario does not formulate subassembly sequences explicitly, rather they are assumed to emerge by searching the space of possible conformational switch designs, *any* subassembly sequence can be realized if it can be encoded by the conformational switch<sup>3</sup>, including the ones which involve subassemblies with non-identical numbers of parts. Finally, although there are only two (imaginary) robot arms (as opposed to many robot arms for parallel assembly), the scenario can simulate the simultaneous construction of several subassemblies since it is globally synchronized based on mating of two subassemblies, rather than completion of a subassembly stage.

## 3.2 Mechanical conformational switch for one-dimensional self-assembly

### 3.2.1 Sliding bar mechanism

The design of the first type of conformational switches, sliding bar mechanisms, is motivated by the “counting device” appearing in [42]. I extended this counting device so that a part can *form* and *destroy* a bond with another part.

As shown in Figure 3-2-a, a part has two bonding sites and a conformational switch is realized with a sliding bar mechanism that connects the two bonding sites. Conformational

---

<sup>2</sup>In fact, the returning does not have to be random if the picking is completely random.

<sup>3</sup>This raises the question of which subassembly sequences can be encoded by a particular conformational switch. Some discussion on this issue is found in Sections 3.5.6 and 3.5.8.

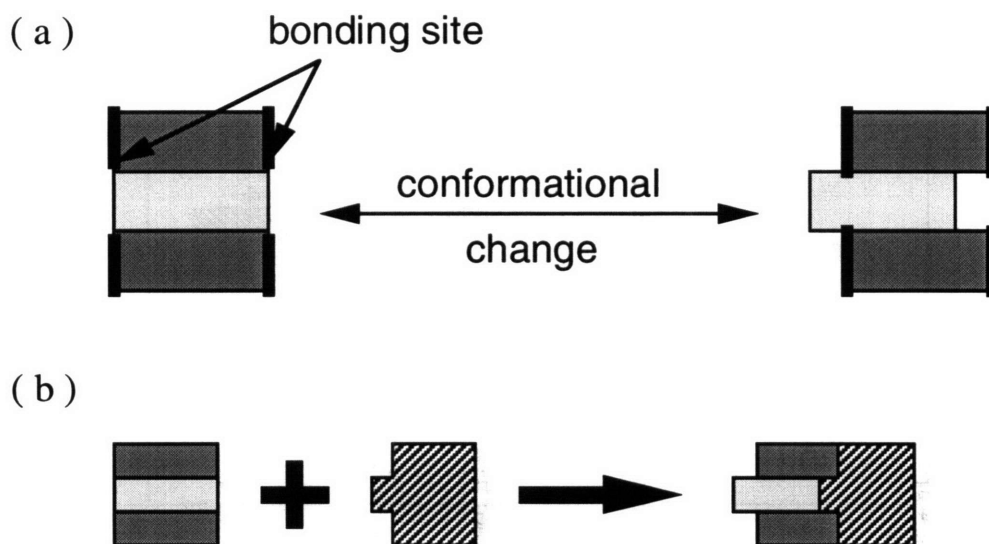


Figure 3-2: One-dimensional conformational switch with a sliding bar mechanism.

change is triggered by interaction with another part (see Figure 3-2-b). Note that this is a conformational switch design for one-dimensional self-assembly, where parts can be assembled in only one direction, say horizontally. In Figure 3-2, therefore, one can place a part on the left or the right of another part, but *not* on the top or bottom. However, the model could easily be extended to the two-dimensional case.

A *bond configuration* is a parameter which describes the shape of a bonding site. It takes a positive value if the corresponding site has convex shape, a negative value if the site is concave, and zero if the site is flat. Examples of bond configurations and the corresponding shape of bonding sites are shown in Figure 3-3.

When bonding sites of two parts meet, they form either 1) a stable bond, 2) an unstable bond, or 3) no bond. The occurrences of these cases depend on the shape of the two bonding sites, or equivalently, the bond configurations of the sites. Let  $a$  and  $b$  be bond configurations of two bonding sites contacting each other. These sites form a stable bond if  $a + b \leq 0$  (complementary), form an unstable bond if  $a + b = 1$  (fairly complementary), and form no bond if  $a + b > 1$  (not complementary). Figure 3-4 shows examples of each of the three cases.

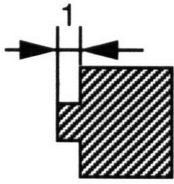
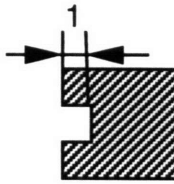

bond site			
bond configuration	+1	-1	0

Figure 3-3: Bond configurations.

An unstable bond induces conformational change of the involved bonding sites, which can propagate over the connected parts via the sliding bar mechanism. After the conformational changes, a stable bond is formed if  $a + b \leq 0$  and no bond is formed if  $a + b \geq 1$ . Also, an existing bond is destroyed if  $a + b \geq 1$  after the conformational changes, which results in detaching of the corresponding parts. Figure 3-5 illustrates an example of such propagation and detaching.

Ambiguous situations may arise when conformational change can propagate in both directions or in no direction, such as the cases shown in Figure 3-6-a. To resolve such ambiguity, we assume an *upstream propagation* priority. As shown in Figure 3-6-b, conformational change propagates downstream (as defined in Figure 3-6), *only* when the upstream direction has a rigid end and downstream has a free end (the bottom picture of Figure 3-6-b). Otherwise, propagation goes upstream (the top and middle picture of Figure 3-6-b).

### 3.2.2 Minus devices

The second type of conformational switches, minus devices, are designed such that they are functionally opposite to the sliding bar mechanisms. Recall as a result of the conformational

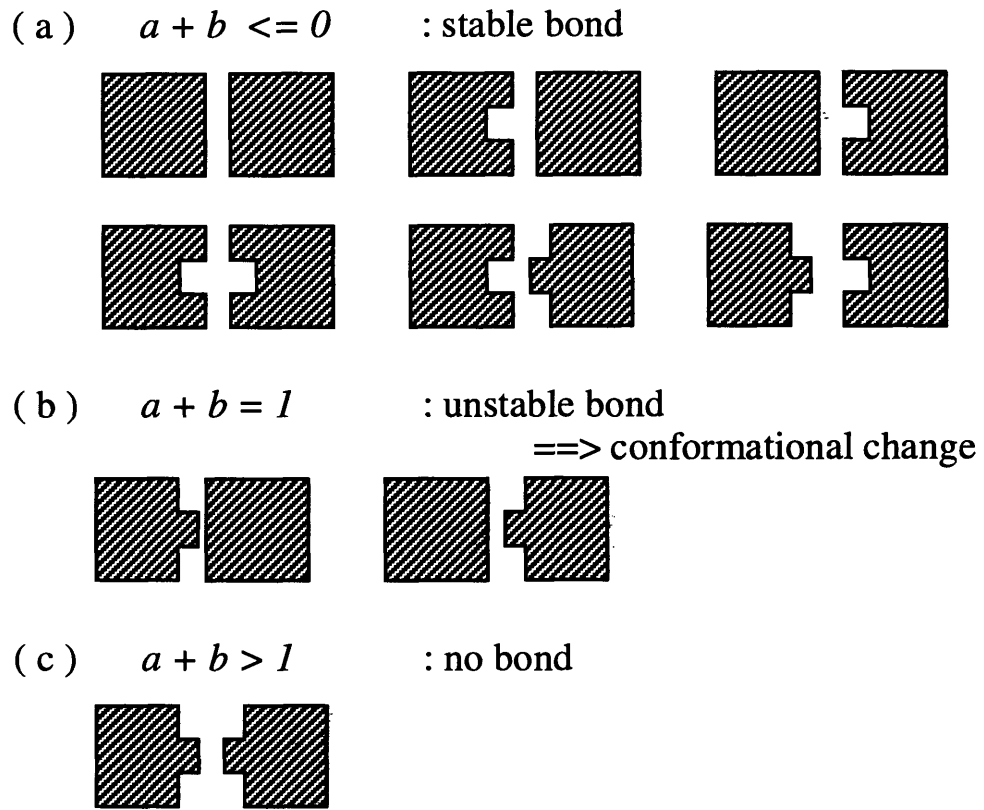


Figure 3-4: Three types of bonding.

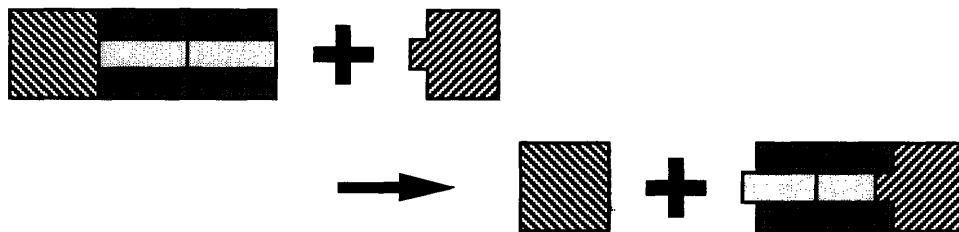
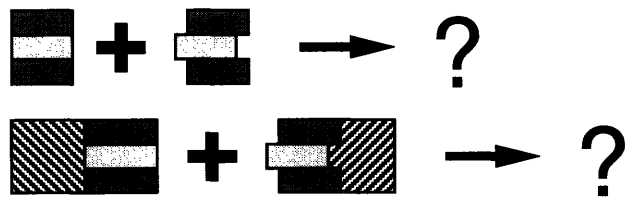


Figure 3-5: Propagation of conformational changes and detaching.

( a ) ambiguous situations



( b ) results by upstream propagation

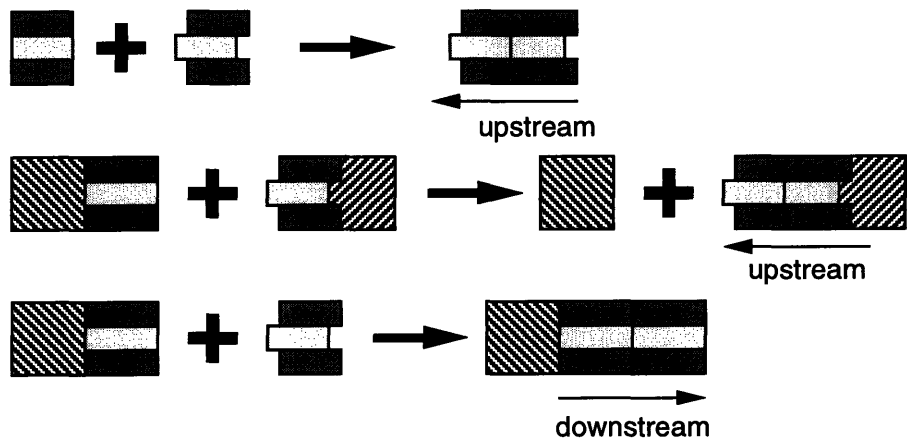


Figure 3-6: Priority to upstream propagation.

change, the sliding bar mechanisms can destroy a bond which is formed *before* the conformational change. In contrast, the minus devices prevents the formation of a bond *before* the conformational change, and allows the formation of a bond *after* the conformational change.

In order to realize this function in a mechanically realistic way, we introduce a new gadget called the *minus device*. Figure 3-7(a) shows the operation of a minus device. It consists of two short sliding bars that constitute the shapes of two bonding sites, and one inner sliding block that connects them. The left and right pictures in Figure 3-7(a) show the minus device *before* and *after* conformational change, respectively. Before conformational change, the right sliding bar *cannot* be pushed in due to the position of the inner sliding block. After conformational change induced by pushing in the left sliding bar, the right sliding bar *can* be pushed in since pushing in the left sliding bar causes the inner block to slide down, leaving space for the right sliding bar to slide left. The device is named “minus” since a sliding bar is being pushed by another part, as opposed to pushing another part as in the case of a sliding bar mechanism. For simplicity, I will often use the abstracted representation of the device, a minus sign with an arrow, shown in Figure 3-7(b). The direction of the arrow indicates that causality of conformational change. A right-pointing arrow indicates the conformational change is induced by pushing the *left* sliding bar and *vice versa*. Note that in the abstract representation, the right sliding bar after conformational change is drawn as the “pushed-in” state, representing that the bar is “free” to be pushed in.

Figure 3-7(c) shows how interaction with another part induces conformational change in a minus device. Note that conformational change realized by a minus device is *unidirectional* and *irreversible*. In the top picture of Figure 3-7(c), for instance, no conformational change is induced if the hatched rectangular part is placed on the *right* of the part with the minus device (unidirectionality). Also, since the minus device is not “spring-loaded”, it is impossible to reverse the conformational change (irreversibility). This implies that once a part changes its conformation and a bond is formed as a result, it will *never* be destroyed, *i.e.* no detaching is possible.

The actual conformational switch design used in the following section consists of a “two-digit” bonding site as shown in Figure 3-8. The two-digit bonding sites are introduced in

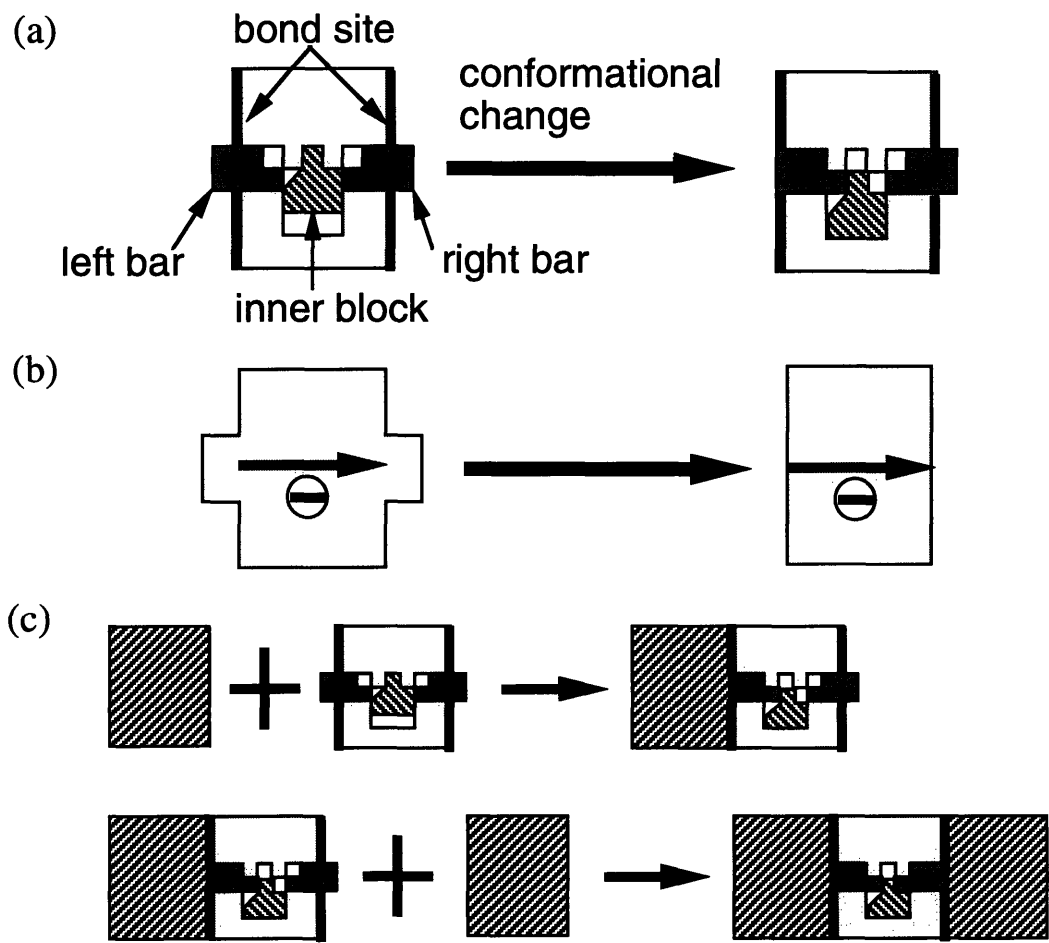


Figure 3-7: One-dimensional conformational switch with a minus device.



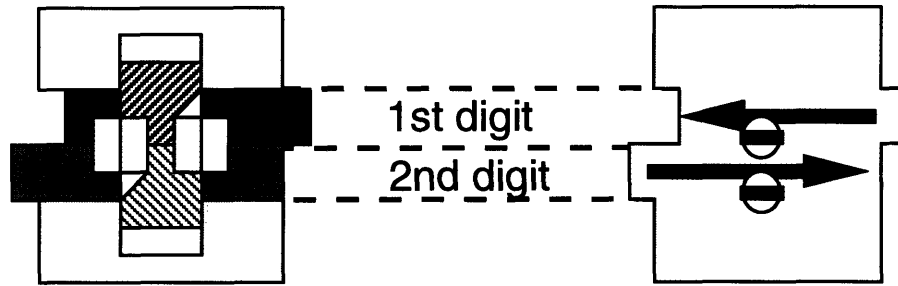


Figure 3-8: One-dimensional conformational switch with “two-digit” bonding sites.

bond site				
bond configuration	( 1, 0 )	(-1, 1 )	( 0, 0 )	(-1, -1 )

Figure 3-9: Examples of bond configurations and the corresponding two-digit bonding site shapes (of the left bonding site).

order to increase the number of possible shapes they can take, which in turn increases the number of subassembly sequences they can encode<sup>4</sup>.

In the case of a two-digit bonding site, a *bond configuration* is a pair of integers  $(a_1, a_2)$ . As in the single-digit case, each component of a bond configuration takes a positive value if the corresponding “digit” of the bonding site has convex shape, a negative value if the the digit is concave, and zero if it is flat. Examples of bond configurations and the corresponding shapes of two-digit bonding sites are shown in Figure 3-9.

The occurrences of the three types of bonding, a stable bond, an unstable bond and no bond, can be defined as follows. Let  $(a_1, a_2)$  and  $(b_1, b_2)$  be bond configurations of

<sup>4</sup>To determine the number of “digits” necessary to encode a given assembly tree is an interesting coding problem and will be discussed in Section 3.5.8.

two bonding sites contacting each other. These sites form a stable bond if they are complementary to each other:

$$a_1 + b_1 \leq 0 \quad \text{and} \quad a_2 + b_2 \leq 0 \quad (3.1)$$

and form an unstable bond if they are fairly complementary:

$$(a_1 + b_1 = 1 \quad \text{and} \quad a_2 + b_2 \leq 0) \quad \text{or} \quad (a_1 + b_1 \leq 0 \quad \text{and} \quad a_2 + b_2 = 1) \quad (3.2)$$

If none of the above apply, the two bonding sites are not complementary and therefore no bond is formed. Figure 3-10 shows some examples of each of the three cases. An unstable bond induces conformational change of the involved bonding sites. After the conformational changes, a stable bond is formed if the condition (3.1) is satisfied. Otherwise, no bond is formed.

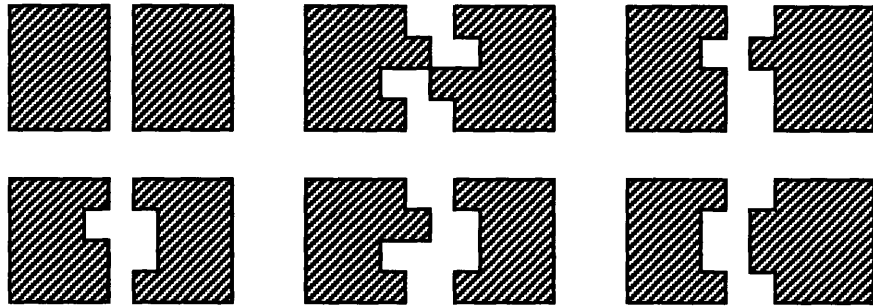
Similar to the single-digit case, ambiguous situations may arise when conformational change can propagate in both directions such as the case shown in Figure 3-11(a). In such cases, I assume propagation goes upstream as defined in Figure 3-11(b) (priority to *upstream propagation*).

### 3.3 Parametric design optimization with Genetic Algorithms

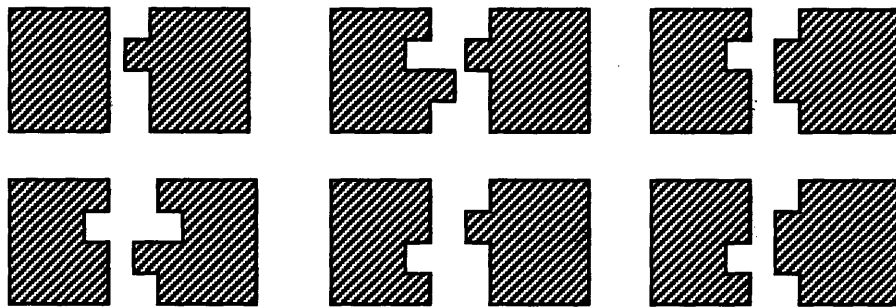
#### 3.3.1 Genetic Algorithms

Genetic algorithms (GAs) are a search technique in which points in the design space are analogous to organisms subject to a process of natural selection, or “survival of the fittest” [21]. GAs model reproduction in populations of an encoded representation of points in design space – genetic “chromosomes” – over generations. In a given generation, the quality of a chromosome (or a point in design space) is measured based on a fitness function, and highly-fit chromosomes have higher chances to be selected for reproduction. Two “parent” chromosomes selected for reproduction are mated through genetic crossover, resulting in two offspring chromosomes which are likely to inherit good “genes” from their parents. Many generations of such selection and mating will produce a highly-fit population

( a ) stable bond



( b ) unstable bond ==> conformational change



( c ) no bond

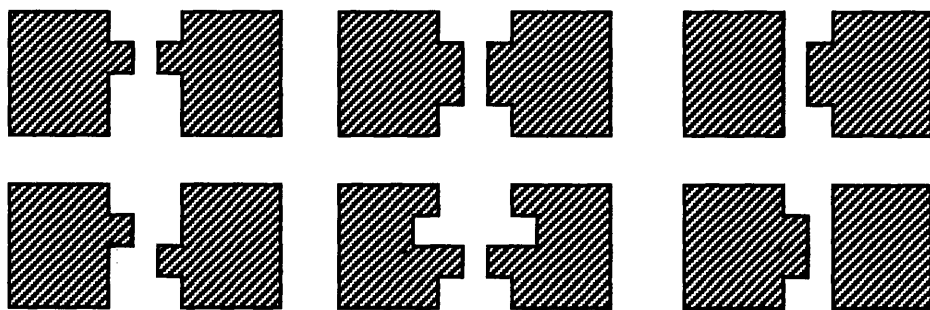
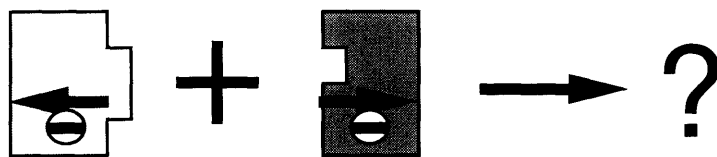


Figure 3-10: Examples of three types of bonding.

(a) ambiguous situation



(b) results by upstream propagation

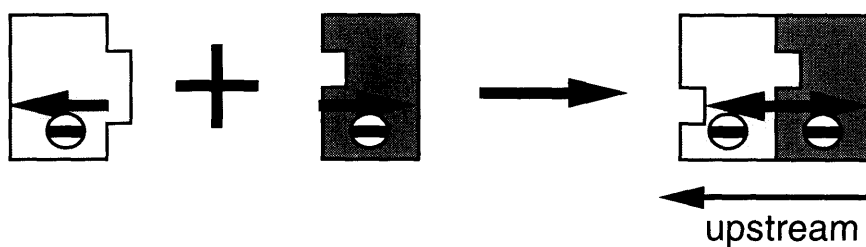


Figure 3-11: Priority to upstream propagation.

of chromosomes, *i.e.* better designs.

### 3.3.2 Problem formulation

I am interested in designing conformational switches that, when randomly assembled, maximize the yield of a desired assembly. More precisely, the problem is stated as follows:

**Given:** the total number of parts, and the number of each kind of part in the bin (in other words, the initial state of the bin), and defect probabilities.

**Find:** the optimal design of conformational switches (and their initial bond configurations) that maximizes the yield of a desired set of parts in the process of the sequential random bin-picking.

The above problem is formulated as search by parameterizing the design of the conformational switches. A genetic algorithm is used to search the parameter space of possible switch designs. It should be noted that I am searching the space of possible conformational switch designs, *not* the space of subassembly sequences. There is no explicit formulation

of subassembly sequences in the above problem. Rather, an optimal subassembly sequence *emerges* due to a particular design of conformational switches that maximizes the yield of the desired assembly.

### 3.3.3 Reinforcement evaluation

A chromosome is evaluated by decoding the bit string representation (genotype) to a part design (phenotype), and by running the sequential random bin-picking simulation with the decoded part design. The fitness of a chromosome (*i.e.* the suitability of conformational switch designs and their initial bond configurations) is simply the number of desired assemblies produced after some number of iterations of the robot bin-picking simulation. In order to reduce stochastic error due to random picking during the simulation, the actual fitness is measured as an average of a pre-specified number of such bin-picking runs.

For further reduction of stochastic error involved in the fitness evaluation, A new scheme called *reinforcement evaluation* is introduced. In this scheme, the best chromosome in the population is evaluated again at the end of each generation. The fitness of the best chromosome is then re-calculated to “reinforce” the evaluation:

$$\bar{f}_{new} = \frac{n \cdot \bar{f}_{old} + f}{n + 1} \quad (3.3)$$

where  $\bar{f}_{old}$  and  $\bar{f}_{new}$  are the fitness values before and after reinforcement,  $f$  is the fitness value obtained by the additional evaluation, and  $n$  is the number of times the chromosome has been evaluated. This ensures that only good chromosomes that are carried over generations have more chances to be evaluated many times, reducing stochastic error of their fitness values. The above reinforcement evaluation scheme is used in the case of three and four part self-assembly described in Section 3.5.

## 3.4 Two part one-dimensional self-assembly with sliding bar mechanisms

This section describes several examples of genetic optimization of sliding bar mechanisms [49]. Unless otherwise specified, the GA in the following examples uses fitness proportionate

(roulette wheel) selection, linear fitness scaling with scaling coefficient = 2.0, an elitist selection scheme, crossover probability = 0.8, and mutation probability = 0.03.

### 3.4.1 Design parameterization

The one-dimensional conformational switch with a sliding bar mechanism is uniquely specified in terms of four parameters: *left\_config*, *right\_config*, *link*, and *bar\_length*. *Left\_config* and *right\_config* are the initial bond configurations of the left bond site and the right bond site, respectively. *Link* is a Boolean variable that specifies the existence of the conformational link (a bar mechanism) in a part. If *link* is TRUE, there is a conformational link between the two bond sites, so they can undergo conformational change. If *link* is FALSE, there is no conformational link, so the bond configurations do not change from their initial values (*i.e.* a solid part). *Bar\_length* is an upper limit on *right\_config* and *left\_config*. In order for a design to be valid, I need  $left\_config, right\_config \leq bar\_length$ . Note that if  $left\_config = right\_config = bar\_length$ , the bar cannot move at all. Also, *bar\_length* is ignored if *link* is FALSE.

A chromosome used in genetic search is a binary string that encodes the above design parameters for all kinds of parts in the bin. For the examples in the next section, two bits are assigned for each of *left\_config* and *right\_config*, and one bit for *link* and *bar\_length*<sup>5</sup>. The location of these bits on a chromosome is shown in Figure 3-12.

For *left\_config* and *right\_config*, the first bit corresponds to the absolute value and the second bit corresponds to sign, with minus being 0 and plus being 1<sup>6</sup>. If  $left\_config = -1$ , for example, the corresponding two bits are 10. Since six bits are necessary for one part, a chromosome that encodes  $n$  kinds of parts has length  $6n$ .

### 3.4.2 Two part sequential assembly

As an preliminary example, a two part sequential assembly is studied, where bin-picking is done deterministically. The assumption here is as follows: two parts, part *A* and part *B*, come in two separate bins, and robot arm #2 is picking a part from each bin in the fixed

---

<sup>5</sup>This implies *bar\_length* can only take values {0, 1}

<sup>6</sup>This implies that *left\_config* and *right\_config* can only take values {-1, 0, 1}.

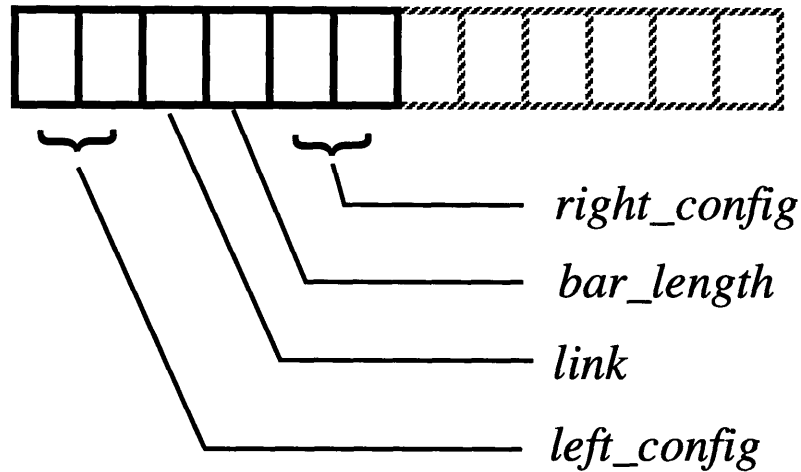


Figure 3-12: Bit assignment of a chromosome.

sequence  $ABAB \dots$ . The robot arm #1 is initially holding a base part (I will call it part  $Z$ ). The assembly is done by pushing parts against each other, followed by release of the part held by arm #2. Since this pushing possibly causes formation and breakage of the existing bond between parts, the assembled parts either stay on arm #1 (no parts detached), or fall off (detaching occurs). The fallen parts are sent to a bin for assembled parts. Arm #2 then goes to the next bin and the process iterates. No defect is assumed in this example. The process is illustrated in Figure 3-13.

The objective is to maximize the yield of the assembly  $AB^7$ . Since the assembly process is deterministic, it suggests the following scenario for the maximum yield:

**Step 1:** Part  $A$  is picked and forms a bond to  $Z$  (i.e.  $Z + A \rightarrow ZA$ ).

**Step 2:** Part  $B$  is picked and forms a bond to  $A$ . Conformational change destroys the bond between  $A$  and  $Z$ , releasing the assembly  $AB$  (i.e.  $ZA + B \rightarrow Z + AB$ )

The evaluation of the switch designs, therefore, needs only two pickings (part  $A$  followed by part  $B$ ). Note that I do not need to take an average of several runs since the process is

---

<sup>7</sup>Note that  $AB$  and  $BA$  are distinct.

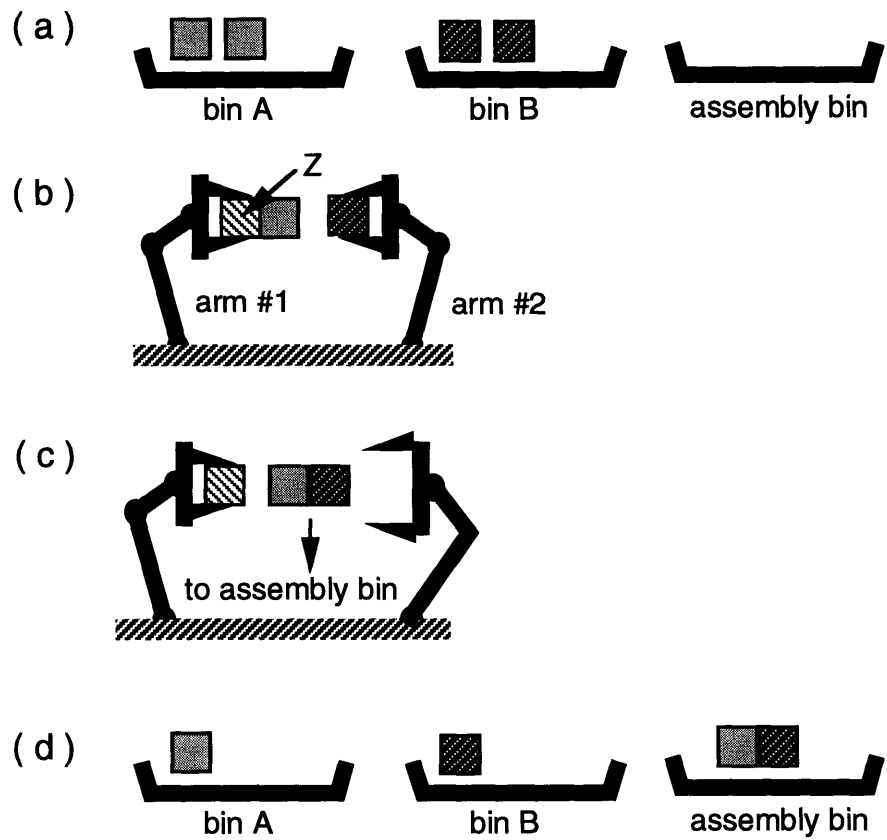


Figure 3-13: Deterministic robot bin-picking.





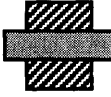
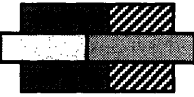


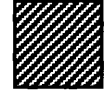





design #	Z	A	B	AB
1	 (0)	 (0, T, 1, 0)	 (1, T, 1, 1)	
2	 (0)	 (0, T, 1, 1)	 (0, F, 0, 0)	
3	 (1)	 (-1, T, 1, 1)	 (0, F, 0, 0)	

Figure 3-14: Best designs for two part, sequential assembly, found by GA.

deterministic. The fitness function is as follows:

$$fitness = \begin{cases} 0 & \text{if } ZA \text{ is not formed at Step 1} \\ 1 & \text{if } ZA \text{ is formed at Step 1 but } AB \text{ is not formed at Step 2} \\ 2 & \text{otherwise} \end{cases} \quad (3.4)$$

The base part  $Z$  is assumed to have only the right bond site and no conformational link, so the size of a chromosome is  $6 \times 2 + 2 = 14$  bits. This means that there are  $2^{14}$  possible designs. Figure 3-14 illustrates the best designs obtained by three GA runs with population size = 10 and number of generations = 5. The numbers below each designs are the corresponding parameters (*left\_config*, *link*, *bar\_length*, *right\_config*)<sup>8</sup>. Only *right\_config* is shown for the base part  $Z$ . Note that all three designs scored the maximum fitness = 2. All the possible designs that score the maximum fitness, found using a depth-first search, are listed in Appendix A.

<sup>8</sup>Note that part B in design #1 is equivalent to a solid part (1, F, 0, 1)




A	B	AB
		
( 1, F, 0, 0 )	( 0, F, 0, 1 )	

Figure 3-15: Best design (part  $A$  : part  $B = 1 : 1$ ) : Design I

### 3.4.3 Two part self-assembly

The second example is a two part self-assembly as described in Section 3.1. The initial bin contains a random mixture of two types of parts, part  $A$  and part  $B$ , and the design objective is to maximize the yield of the assembly  $AB$ . The total number of parts in the initial bin is fixed at 50 and the number of  $AB$ 's in the bin is counted after 50 iterations of Steps 1-3 in Section 3.1. As in the previous example, no defect is assumed. The fitness of a chromosome is the average count of  $AB$ 's over 50 such runs. In the GA runs described below, the population size is 30 and the number of generations is 10.

Figure 3-15 shows the best design (fitness = 9.76) when the initial fraction of part  $A$ 's and part  $B$ 's is 1:1 (*i.e.* 25 part  $A$ 's and 25 part  $B$ 's). Let us call it Design I. As easily seen from the figure, the result is as expected: only  $A + B \rightarrow AB$ <sup>9</sup> occurs.

Figure 3-16-a shows the best design (fitness = 6.7) when the initial fraction of part  $A$ 's and part  $B$ 's is 4:1 (*i.e.* 40 part  $A$ 's and 10 part  $B$ 's). Let us call it Design II. All possible assemblies with this design are illustrated in Figure 3-16-b, where  $A'$  represents part  $A$  after conformational change. Note that not only  $A + B \rightarrow A'B$  (reaction 2), but also  $A + A \rightarrow A'A$  (reaction 1) and  $A' + A \rightarrow A'A$  (reaction 3), are possible. Once  $A'A$  is formed, it can also be bound to  $B$  to produce  $A'B$  (*i.e.*  $A'A + B \rightarrow A' + A'B$ : reaction 7). The reactions 1, 3 and 5 help to decrease the total number of part  $A$ 's in the bin, and

<sup>9</sup>It is implicitly assumed the operator '+' is not commutative. In particular,  $B + A$  is not possible in this case.

in turn, to increase the chance of part  $B$ 's being picked. If a part  $B$  is picked, it can bind to any of  $A$ ,  $A'$ , and  $A'A$  (by reactions 2, 4, and 7). Also, once  $A'B$  is formed, it can never be destroyed<sup>10</sup>. The overall yield of  $A'B$ 's, therefore, is better than that from only  $A + B \rightarrow AB$ . For comparison, the typical fitness of the design in Figure 3-15, when applied to the same situation (*i.e.* part  $A$  : part  $B = 4 : 1$ ), is 5.5. The next section describes a comparison of these two designs based on expected yield.

### 3.4.4 Rate equation analyses of two part self-assembly

Given the part designs and their possible reactions, one can formulate a discrete-time rate equation similar to the ones found in [53, 27]. The following recurrence describes the self-assembly process described in Section 3.1:

$$\mathbf{n}(t + 1) = \mathbf{n}(t) + \mathbf{A}\mathbf{p}(t) \quad (3.5)$$

where  $\mathbf{n}(t)$  is a vector of the numbers of each possible subassembly 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. Since only one reaction  $A + B \rightarrow AB$  is possible for Design I and no defect is assumed,  $\mathbf{n}(t)$ ,  $\mathbf{A}$  and  $p(t)$  (scalar in this case) are defined as follows:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_{AB}(t) \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \quad p(t) = \frac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \quad (3.6)$$



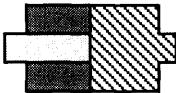
where  $n_A(t)$ ,  $n_B(t)$  and  $n_{AB}(t)$  are the numbers of part  $A$ , part  $B$  and assemblies  $AB$  at iteration  $t$ , respectively, and  $s(t) = n_A(t) + n_B(t) + n_{AB}(t)$ .

In the case of Design II, the possible reactions are:

---

<sup>10</sup>Note that reverse of reaction 7  $A' + A'B \rightarrow A'A + B$  is *not* possible due to the upstream priority rule discussed in Section 3.2.1.

( a ) best design of part A and part B : Design II

A	B	A'B
		
( 0, T, 1, 1 )	( 0, F, 1, 1 )	

( b ) possible combinations of assemblies

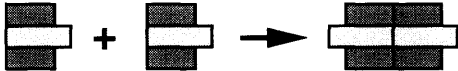
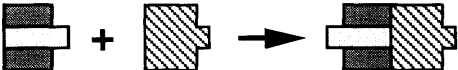





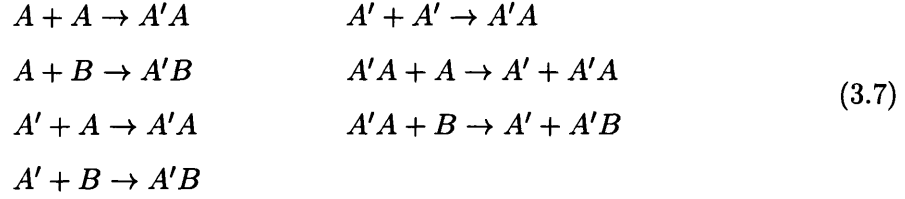
No.	assembly
1	 : A + A --> A'A
2	 : A + B --> A'B
3	 : A' + A' --> A'A
4	 : A' + B --> A'B
5	 : A' + A' --> A'A
6	 : A'A + A --> A' + A'A
7	 : A'A + B --> A' + A'B

Figure 3-16: Best design (part A : part B = 4 : 1)



Corresponding  $\mathbf{n}(t)$ ,  $\mathbf{A}$  and  $\mathbf{p}(t)$  are, therefore, defined as follows:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_{A'}(t) \\ n_{A'A}(t) \\ n_{A'B}(t) \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} -2 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & -1 & -2 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$\mathbf{p}(t) = \begin{pmatrix} \frac{n_A(t) \cdot \{n_A(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{A'}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{A'}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{A'}(t) \cdot \{n_{A'}(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{A'A}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{A'A}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \tag{3.9}$$

For Design I and Design II, equation 3.5 is solved numerically with the two initial conditions discussed in Section 3.4.3. Figure 3-17-a and Figure 3-17-b are the solution for Design I and Design II with initial condition  $\mathbf{n}(0) = (25, 25, 0)'$ , and  $\mathbf{n}(0) = (25, 25, 0, 0, 0)'$ ,

respectively. Note that for *both* Design I and Design II,  $n_A, n_B \rightarrow 0$  and  $n_{AB}^{11} \rightarrow 25$  as  $t \rightarrow \infty$ . The yield of the desired assembly  $AB$  is compared in Figure 3-18. It shows Design I is consistently better than Design II for  $t \in \{0, 500\}$ , and the difference in the yield is maximum at  $t \simeq 150$ . At  $t = 50$ ,  $n_{AB}(t)$  for Design I and Design II are 9.8914 and 9.6590, respectively.

Similar analysis is done with the initial conditions  $\mathbf{n}(0) = (40, 10, 0)'$  for Design I, and  $\mathbf{n}(0) = (40, 10, 0, 0, 0)'$  for Design II. As shown in Figures 3-19-a and 3-19-b, Asymptotic behaviors of  $n_B(t)$  and  $n_{AB}(t)$  are similar in Design I and Design II. In Design II, however,  $n_A(t) \rightarrow 0$  quickly whereas in Design I,  $n_A(t) \rightarrow 30$  as  $t \rightarrow \infty$ . This drop of  $n_A(t)$  and increase of  $n_{A'A}(t)$  results in decrease of total number of parts in the bin  $s(t)$ , which is clearly shown in Figure 3-19-b. This supports the qualitative argument in Section 3.4.3. Comparison of the yield in Figure 3-20 indicates Design II is consistently better than Design I for  $t \in \{0, 500\}$ , and the difference in the yield is maximum at  $t \simeq 100$ . At  $t = 50$ ,  $n_{AB}(t)$  for Design I and Design II are 5.7961 and 6.3230, respectively.

### 3.4.5 Two part self-assembly with a dummy part

This example is the same as the previous example except that a dummy part  $C$  is mixed in the initial bin. The dummy part  $C$  is not involved in the desired assembly  $AB$ , but is expected somehow to help part  $A$ 's and part  $B$ 's assemble to  $AB$  (as an enzyme does in viral assembly).

Figure 3-21 shows the best design (fitness = 2.74) where initially  $A : B : C = 8 : 1 : 1$  (*i.e.* 40 part  $A$ 's, 5 part  $B$ 's and 5 part  $C$ 's). Let us call this Design I. The designs of part  $A$  and part  $B$  are identical to the designs of part  $A'$  and part  $B$  in Figure 3-16, respectively, and part  $C$  cannot bond to any of the possible assemblies. In other words, part  $C$  does not play any role in the assembly of  $AB$ .

This, however, is not the case if the fraction of part  $C$ 's is larger. Figure 3-22 shows Design II, the best design (fitness = 2.84) where  $A : B : C$  is 11 : 3 : 11 (*i.e.* 22 part  $A$ 's, 6 part  $B$ 's and 22 part  $C$ 's). Part  $A$  and part  $B$  are the same as in the design of Figure 3-16. This time, however, part  $C$  can bind to part  $A$ , part  $B$ , and other assemblies. During the

---

<sup>11</sup>denoted  $n_{A'B}$  in Design II.

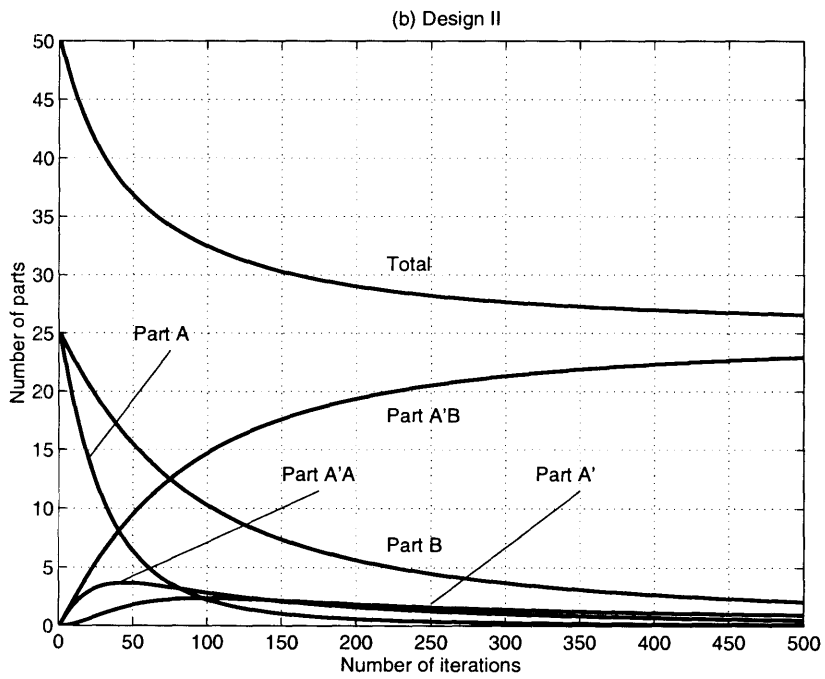
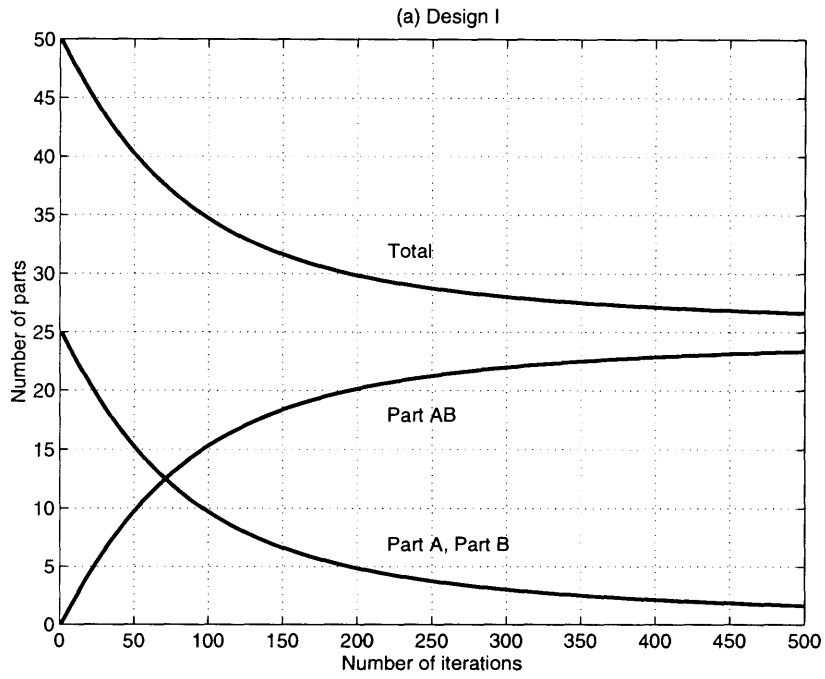


Figure 3-17: Solution of Equation 3.5 ( $A : B = 1 : 1$ ).

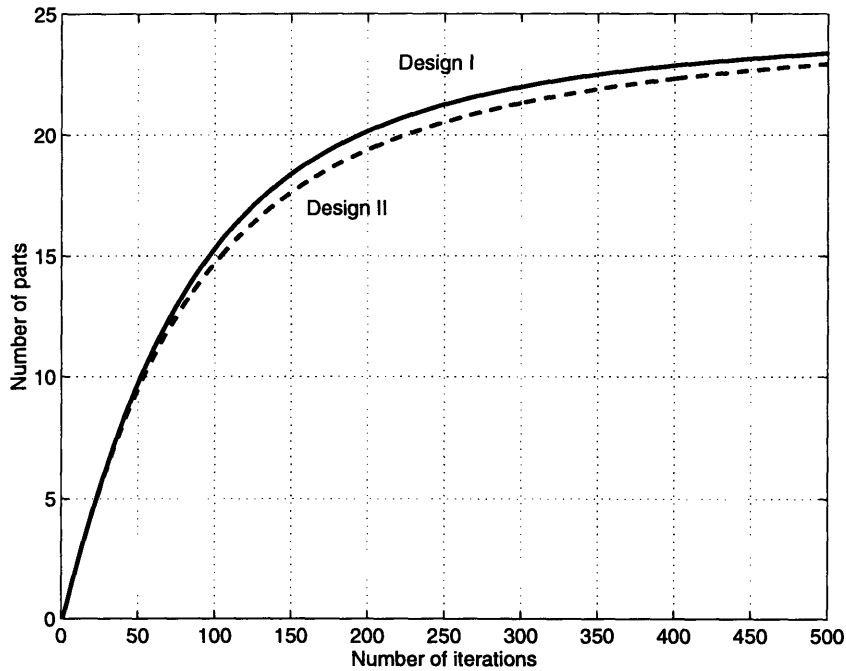


Figure 3-18: Yield computed by Equation 3.5 ( $A : B = 1 : 1$ ).

GA run, it was observed that part  $A$ 's and part  $C$ 's form a long chain such as  $A'C \cdots CA$ , which seems to help to increase the chance of part  $B$ 's being picked.

### 3.4.6 Rate equation analyses of two part self-assembly with a dummy part

The rate equations for Design I are relatively simple. For Design II, however, the number of possible subassemblies and the number of possible reactions can be very large since part  $C$ 's can form a subassembly  $C_n$  (an  $n$  concatenation of part  $C$ 's) for any positive integer  $n$ . To get around this problem, I simply do not distinguish part  $C_n$  and part  $C_m$  for any positive integers  $n$  and  $m$ . This reduces the number of subassemblies down to 11. The resulting rate equations are still useful since the desired assembly  $A'B$  does not contain part  $C$ 's. Also, there is no need to keep track of the number of each  $C_n$ 's. The derivation of the rate equations is found in Appendix B.

Figure 3-23 shows the comparison of the yield by Design I and by Design II with the initial conditions  $\mathbf{n}(0) = (40, 5, 5, 0, 0)'$  and  $\mathbf{n}(0) = (40, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0)'$ , respectively.



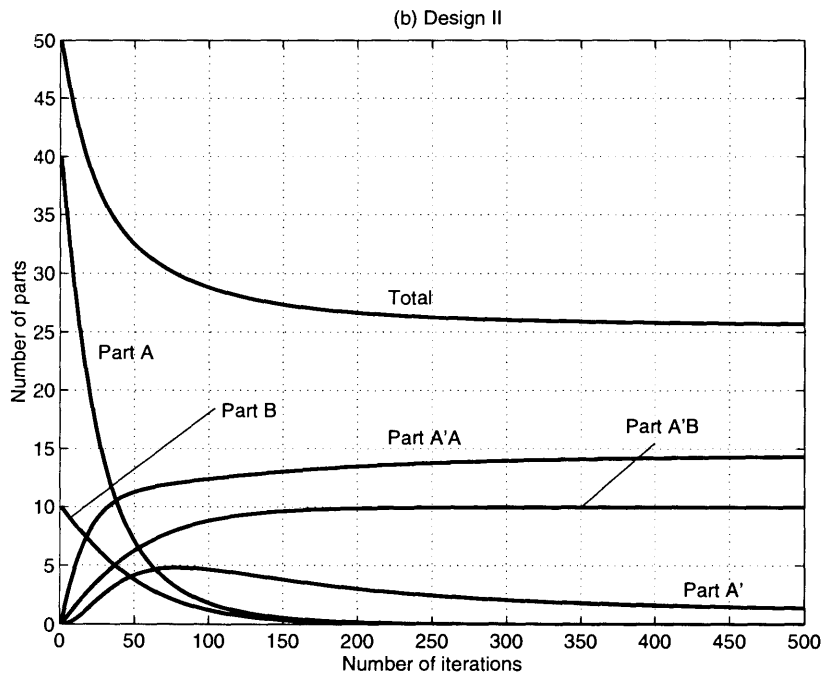
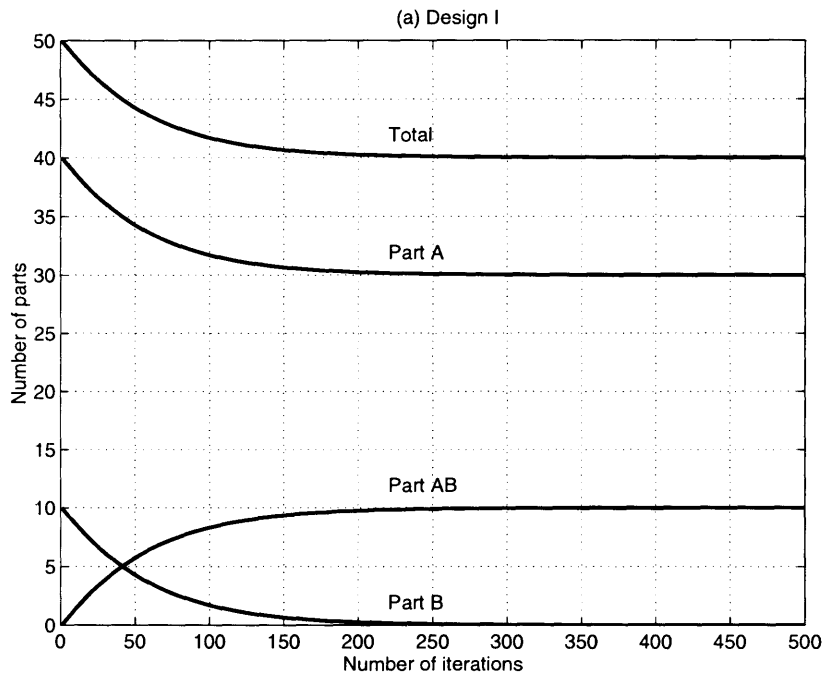


Figure 3-19: Solution of Equation 3.5 ( $A : B = 4 : 1$ ).

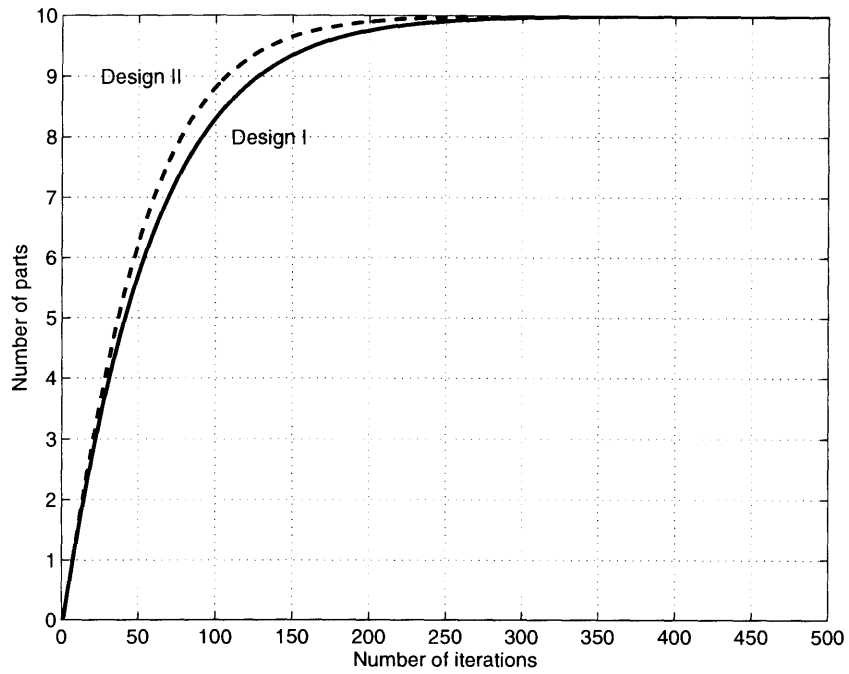


Figure 3-20: Yield computed by Equation 3.5 ( $A : B = 4 : 1$ ).

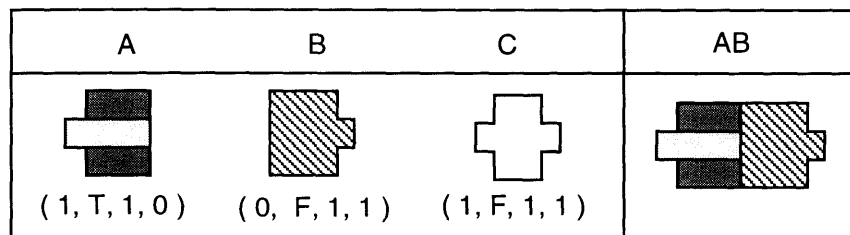


Figure 3-21: Best design ( $A : B : C = 8 : 1 : 1$ ) : Design I.





A	B	C	AB
			
(0, T, 1, 1)	(0, F, 1, 1)	(0, F, 0, 0)	

Figure 3-22: Best design ( $A : B : C = 11 : 3 : 11$ ) : Design II.

It is observed that as  $t \rightarrow \infty$ , the yield of Design I converges to 5 (maximum possible) and the yield of Design II goes to 4.5. At  $t = 50$ , however, the yield of Design I is 3.1260, whereas it is 3.1551 for Design II. In other words, the numerical analysis indicates that at  $t = 50$ , Design II is slightly better in yield, in contrast to the simulation results. A similar trend is observed for the second case where  $\mathbf{n}(0) = (22, 6, 22, 0, 0)'$  for Design I and  $\mathbf{n}(0) = (22, 6, 22, 0, 0, 0, 0, 0, 0, 0)'$  for Design II (Figure 3-24). Even though Design I is asymptotically better in yield, Design II barely wins at  $t = 50$ :  $n_{A'B}(50) = 2.1148$  for Design I and  $n_{A'B}(50) = 2.1709$  for Design II. This matches the simulation result, but the difference in yield is marginal.

### 3.5 Subassembly generation in multi-part one-dimensional self-assembly with minus devices

This section describes two examples of genetic optimization of one-dimensional conformational switches with minus devices [50]. The GA in the following examples uses a crowding population [21] with 10% replacement per generation, based on hamming distance between chromosomes, fitness proportionate (roulette wheel) selection [21], linear fitness scaling [21] with scaling coefficient = 2.0, and unless otherwise specified, crossover probability = 0.9 and mutation probability = 0.03. Also, reinforcement evaluation is performed as described in Section 3.3.3.

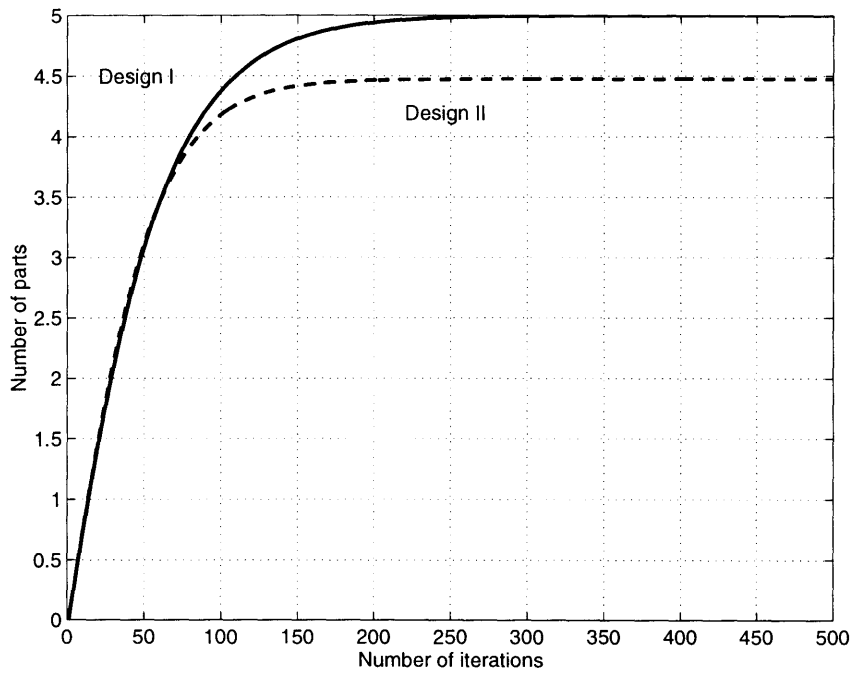


Figure 3-23: Yield computed by Equation 3.5 ( $A : B : C = 8 : 1 : 1$ ).

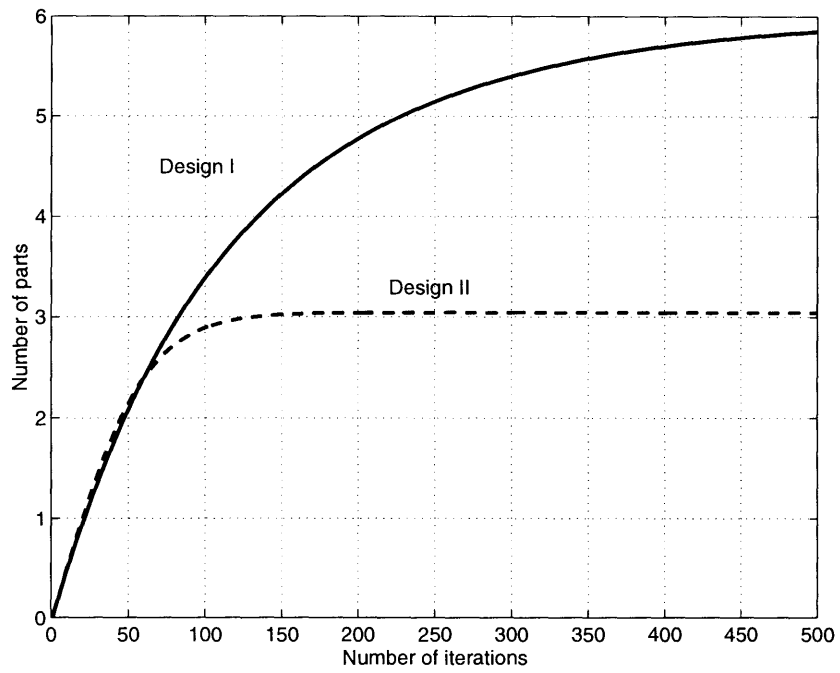


Figure 3-24: Yield computed by Equation 3.5 ( $A : B : C = 11 : 3 : 11$ ).

### 3.5.1 Design parameterization

The following four parameters uniquely specify the one-dimensional conformational switch with two minus devices and two-digit bonding 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 conformational links (minus devices) in a part, each of which takes one of the values *LEFT*, *RIGHT* or *NONE*. If *upper\_link* is *LEFT*, there is a conformational link between the “first digits” of the bond sites, with the direction pointing to the left bond site. If *upper\_link* is *RIGHT*, there is a conformational link between the first digits of the bond sites pointing to the right bond site. If *upper\_link* is *NONE*, there is no conformational link that connects the first digits of the bond sites, so the first digits cannot undergo any conformational change. Similarly, *lower\_link* specifies the existence of the conformational link between the second digits of the bond sites. Note that the conformational links are independent of each other, so the second digits can change their conformation, for instance, even though the first digits cannot.

A chromosome used in genetic search is a binary string that encodes the above design parameters for all kinds of parts in the bin. For the examples in the next section, two bits are assigned to each part of bond configuration<sup>12</sup>, therefore each of *left\_config* and *right\_config* occupies four bits. In addition, two bits are used for each of *upper\_link* and *lower\_link*. The location of these bits on a chromosome is shown in Figure 3-25.

For each part of *left\_config* and *right\_config*, the first bit corresponds to sign, with plus being 0 and minus being 1, and the second bit corresponds to the absolute value<sup>13</sup>. If *left\_config* = (1,0), for example, the corresponding four bits are 0100 or 0110. The first bit for *upper\_link* and *lower\_link* represents the direction of the conformational link, with right ( $\rightarrow$ ) being 0 and left ( $\leftarrow$ ) being 1. The second bit represents the existence of the link. The bit is 1 if the link exists, and 0 if there is no link. The first bit is ignored if the second bit is 0. For instance, the corresponding two bits for *lower\_link* = *NONE* is either 10 or 00. Figure 3-26 shows an example of a parameter encoding of a part. Since twelve bits are necessary for one part, a chromosome that encodes  $n$  kinds of parts has length  $12n$ .

---

<sup>12</sup>Recall a bond configuration is a pair of integer  $(a_1, a_2)$ .

<sup>13</sup>This implies that each part of *left\_config* and *right\_config* can only take values  $\{-1, 0, 1\}$ .

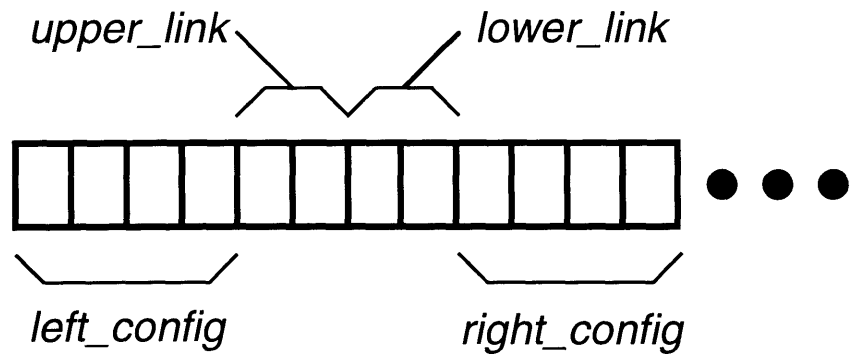


Figure 3-25: Bit assignment of a chromosome.

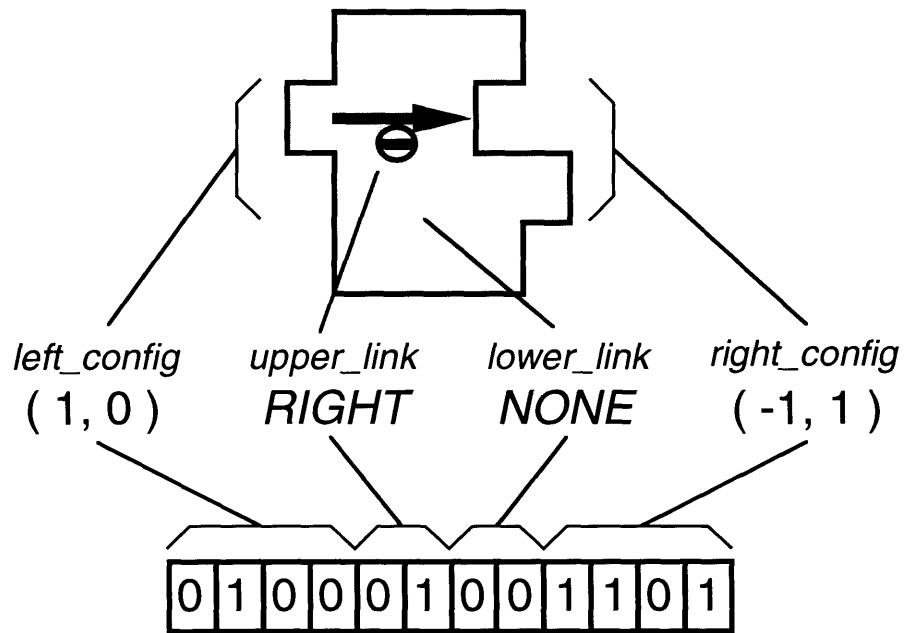


Figure 3-26: Example of parameter encoding of a part.

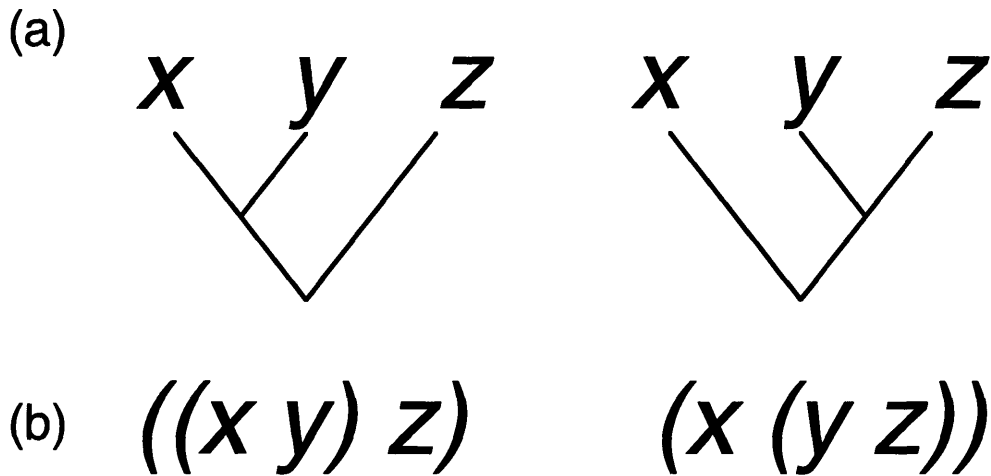


Figure 3-27: Representations of subassembly sequences: (a) binary tree representation, and (b) list representation.  $x$ ,  $y$  and  $z$  are subassemblies.

### 3.5.2 Representaion of subassembly sequences

Before proceeding, let us define our notion of subassembly sequences in one-dimensional assembly. I define a *subassembly* to be a set of one or more parts connected together. In particular, a part is a subassembly. A *subassembly sequence* is a sequence in which *two* subassemblies are put together to produce a final assembly. According to the above definition, *any* fixed (non-ambiguous) subassembly sequence of a one-dimensional assembly can be represented uniquely by a binary tree (Figure 3-27(a)) or by a list representation (Figure 3-27(b))<sup>14</sup>

It is, however, often the case that the assembly sequence encoded by a conformational switch design is ambiguous, or under-specified. I use the notation  $\{u\}$  if (and only if) the subassembly sequence to build a subassembly  $u$  is not specified. In the case where  $u = xyz$ , for instance,  $\{xyz\}$  indicates that the three subassemblies  $x$ ,  $y$  and  $z$  can be put together in any order, i.e. either  $((xy)z)$  or  $(x(yz))$ .

<sup>14</sup>Since I am dealing with one-dimensional assembly, the above binary tree representation can specify *both* a final assembly *and* the order of assembly, whereas in mechanical assembly an *assembly tree* usually specifies only the order of assembly.

### 3.5.3 Three part one-dimensional self-assembly

The first example is a three part one-dimensional self-assembly as described in Section 3.1. The initial bin contains a random mixture of three types of parts, part  $A$ , part  $B$  and part  $C$ , and the design objective is to maximize the yield of the assembly  $ABC$ . The number of  $ABC$ 's in the bin is counted after 700 iterations of Steps 1-3 in Section 3.1. At each evaluation of a chromosome, an average is taken for the count of  $ABC$ 's over 50 such runs. The GA runs described in this section have population size of 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<sup>15</sup>. Note that there are only two possible subassembly sequences in this example:  $((AB)C)$  and  $(A(BC))$ .

Figure 3-28 shows the best designs of conformational switches obtained from GA runs with  $\mathbf{n}_0 = (10, 10, 10)$ , and with (a)  $\mathbf{q} = (0.0, 0.0)$ , (b)  $\mathbf{q} = (0.2, 0.0)$  and (c)  $\mathbf{q} = (0.0, 0.2)$ . In all three cases, the parts are designed such that *only*  $ABC$  can form through random mating (*e.g.*  $CAB$  is not possible). During assembly of  $ABC$ , however, no conformational links are actually used, *i.e.* no parts undergo conformational changes. This implies that none of the three designs specifies a fixed subassembly sequence: an  $ABC$  can assemble in any of the two possible sequences,  $((AB)C)$  or  $(A(BC))$ . In other words, these conformational switch designs encode  $\{ABC\}$ .

On the other hand, A non-ambiguous subassembly sequence emerges in the case where there are more part  $B$ 's than part  $A$ 's and part  $C$ 's. Figure 3-29 shows the resulting switch designs in the case  $\mathbf{n}_0 = (10, 20, 10)$ . 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  $C$  (see Figures 3-29(a) and 3-29(c)). The formation of an assembly  $ABC$ , therefore, takes place through the following two-step "reactions":

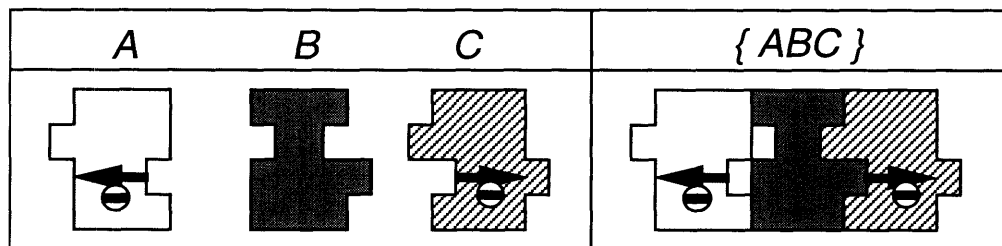



---

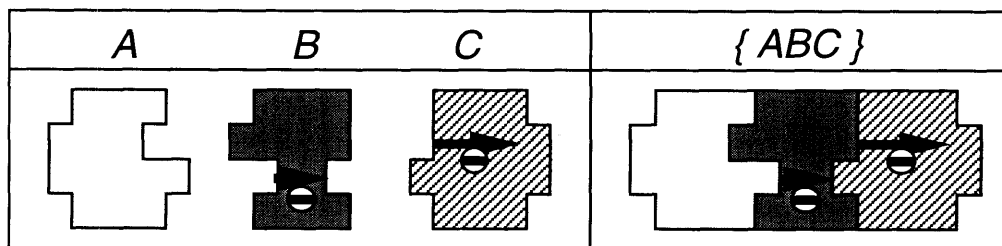
<sup>15</sup>I assume defect probability of a bond depends only on the parts associated to the bond. In particular, I assume  $q_{AB} = q_{A(BC)}$  and  $q_{BC} = q_{(AB)C}$ .



(a)  $\mathbf{q} = (0.0, 0.0)$ ; fitness = 9.99;  $n = 1$



(b)  $\mathbf{q} = (0.2, 0.0)$ ; fitness = 8.09;  $n = 1$



(c)  $\mathbf{q} = (0.0, 0.2)$ ; fitness = 8.01;  $n = 1$

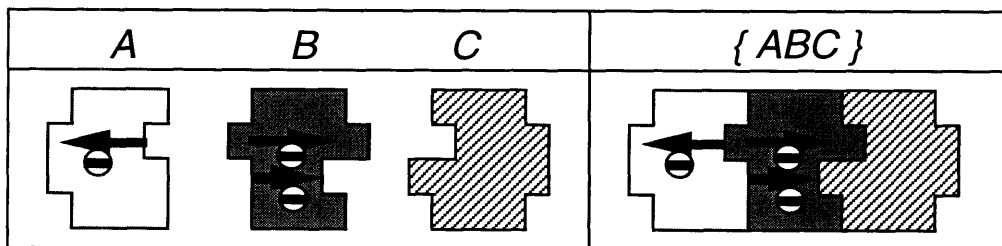


Figure 3-28: Best designs with  $\mathbf{n}_0 = (10, 10, 10)$ .

$\mathbf{q} \backslash \mathbf{n}_0$	(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 ) )

Table 3.1: Summary of the results: three part self-assembly.

where  $B'$  represents a part  $B$  after conformational change. Since no other reactions are possible, an  $ABC$  assembles *only* in the fixed subassembly sequence  $(A(BC))$ . On the other hand,  $((AB)C)$  is encoded in the best design with  $\mathbf{q} = (0.2, 0.0)$  as shown in Figure 3-29(b). In this case, a part  $C$  can bind to a part  $B$  *only after* the part  $B$  changes its conformation, which is triggered by binding of a part  $A$ :



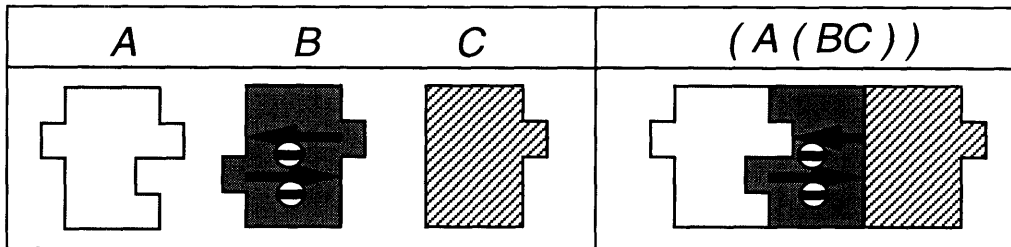
Note that only one conformational link is actually used during the assembly of  $ABC$  in both cases.

The results of GA runs with  $\mathbf{n}_0 = (20, 20, 10)$  are shown in Figure 3-30. The best designs encode  $((AB)C)$  for  $\mathbf{q} = (0.0, 0.0)$  and  $\mathbf{q} = (0.2, 0.0)$ , and  $(A(BC))$  for  $\mathbf{q} = (0.0, 0.2)$ . The summary of these nine GA runs are shown in Figure 3.1.

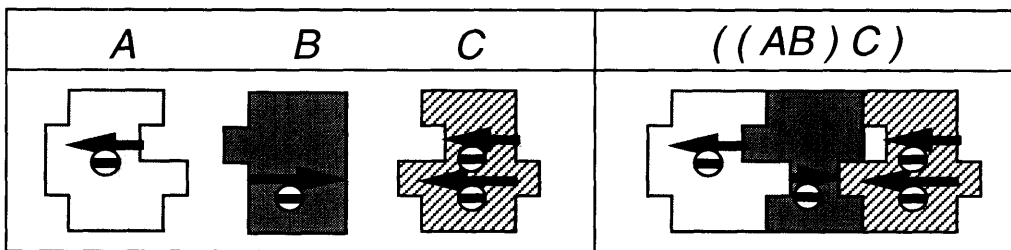
### 3.5.4 Rate equation analyses of three part self-assembly

Discrete-time rate equation analyses are done in order to understand the emergence of a particular subassembly sequence in the above example. The rate equation formulation for defect-free self-assembly in Section 3.4.4 is generalized to incorporate the effect of defects in assembly. Even with non-zero defect probabilities, the rate equation is in the form of

(a)  $\mathbf{q} = (0.0, 0.0)$ ; fitness = 9.52;  $n = 1$



(b)  $\mathbf{q} = (0.2, 0.0)$ ; fitness = 7.84;  $n = 9$



(c)  $\mathbf{q} = (0.0, 0.2)$ ; fitness = 8.07;  $n = 2$

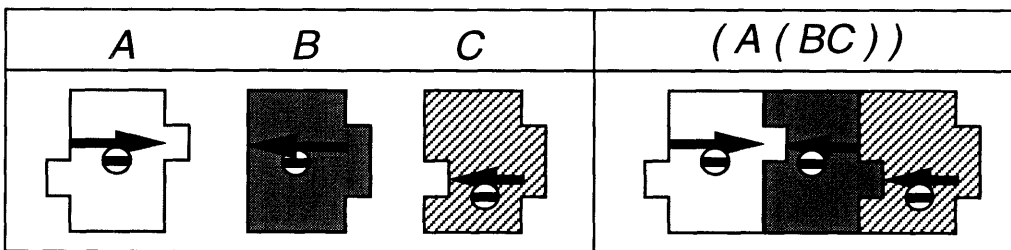
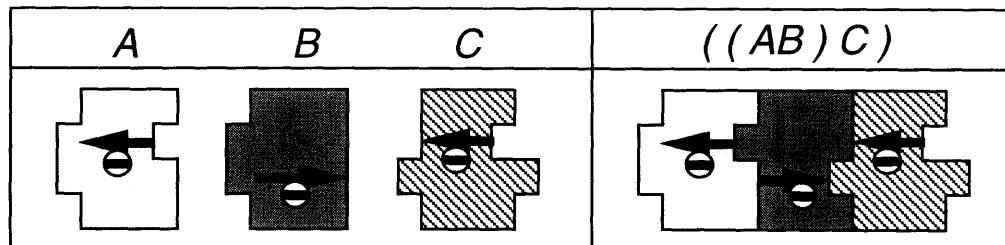
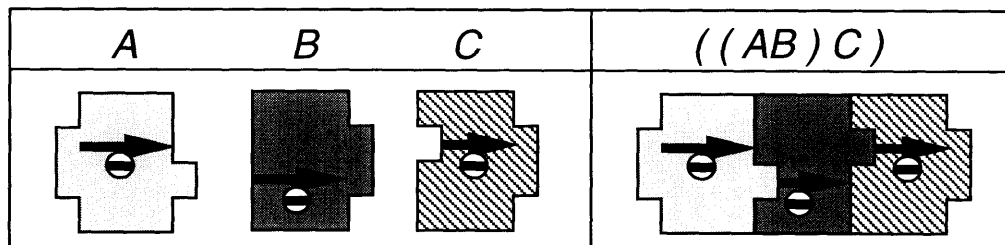


Figure 3-29: Best designs with  $\mathbf{n}_0 = (10, 20, 10)$ .

(a)  $\mathbf{q} = (0.0, 0.0)$ ; fitness = 10.00;  $n = 11$



(b)  $\mathbf{q} = (0.2, 0.0)$ ; fitness = 9.98;  $n = 2$



(c)  $\mathbf{q} = (0.0, 0.2)$ ; fitness = 8.21;  $n = 1$

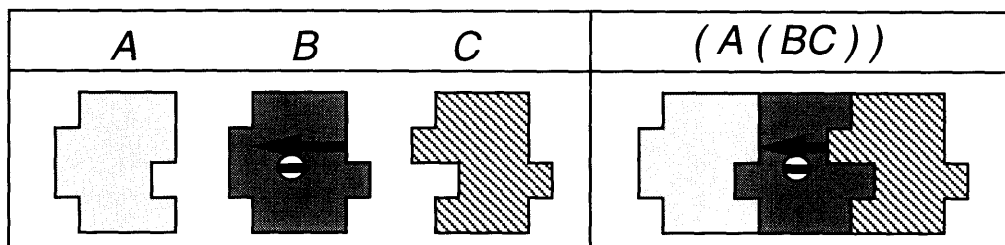


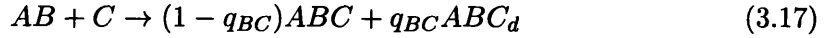
Figure 3-30: Best designs with  $\mathbf{n}_0 = (20, 20, 10)$ .

Equation 3.5, except that the vector  $\mathbf{n}(t)$  includes the number of defective subassemblies at iteration  $t$ .

The following two reactions among parts  $A$ ,  $B$  and  $C$  are necessary and sufficient to produces an assembly  $ABC$  in the subassembly sequence  $((AB)C)$ <sup>16</sup>:



The above reactions can be interpreted as “an  $A$  and a  $B$  yield an  $AB$  with probability 1, and an  $AB$  and a  $C$  yield an  $ABC$  with probability 1.” Assuming the defective subassemblies cannot be incorporated into the subsequent subassemblies, the reactions (3.14), (3.15) can be generalized to non-zero defect probabilities as follows:



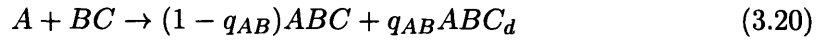
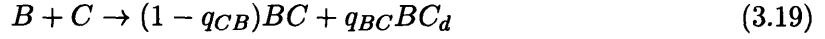
where  $AB_d$  and  $ABC_d$  denote a defective  $AB$  and a defective  $ABC$ ; and  $q_{AB}$  and  $q_{BC}$  are defect probabilities of bonding between  $A$  and  $B$ , and between  $B$  and  $C$ , respectively. The corresponding  $\mathbf{n}(t)$ ,  $\mathbf{A}$  and  $\mathbf{p}(t)$  are, therefore, defined as follows:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_C(t) \\ n_{AB}(t) \\ n_{AB_d}(t) \\ n_{ABC}(t) \\ n_{ABC_d}(t) \end{pmatrix}, \mathbf{A} = \begin{pmatrix} -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 1 - q_{AB} & -1 \\ q_{AB} & 0 \\ 0 & 1 - q_{BC} \\ 0 & q_{BC} \end{pmatrix}, \mathbf{p}(t) = \begin{pmatrix} \frac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{AB}(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \quad (3.18)$$

<sup>16</sup>Conformational change is ignored in the notation here.

where  $n_A(t)$ ,  $n_B(t)$ ,  $n_C(t)$ ,  $n_{AB}(t)$ ,  $n_{AB_d}(t)$ ,  $n_{ABC}(t)$  and  $n_{ABC_d}(t)$  are the numbers of  $A$ ,  $B$ ,  $C$ ,  $AB$ ,  $AB_d$ ,  $ABC$  and  $ABC_d$  at iteration  $t$ , respectively, and  $s(t)$  is the sum of all the parts of  $\mathbf{n}(t)$ .

Similarly, in the case of  $(A(BC))$ , the possible generalized reactions are:



and the corresponding  $\mathbf{n}(t)$ ,  $\mathbf{A}$  and  $\mathbf{p}(t)$  are:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_C(t) \\ n_{BC}(t) \\ n_{BC_d}(t) \\ n_{ABC}(t) \\ n_{ABC_d}(t) \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \\ -1 & 0 \\ 1 - q_{BC} & -1 \\ q_{BC} & 0 \\ 0 & 1 - q_{AB} \\ 0 & q_{AB} \end{pmatrix}, \mathbf{p}(t) = \begin{pmatrix} \frac{n_B(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_A(t) \cdot n_{BC}(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \quad (3.21)$$

Since both  $((AB)C)$  and  $(A(BC))$  are possible in the case of  $\{ABC\}$ , the rate equations for  $\{ABC\}$  are constructed by simply merging (3.18) and (3.21) together:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_C(t) \\ n_{AB}(t) \\ n_{AB_d}(t) \\ n_{BC}(t) \\ n_{BC_d}(t) \\ n_{ABC}(t) \\ n_{ABC_d}(t) \end{pmatrix}, \mathbf{A} = \begin{pmatrix} -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 1 - q_{AB} & 0 & 0 & -1 \\ q_{AB} & 0 & 0 & 0 \\ 0 & 1 - q_{BC} & -1 & 0 \\ 0 & q_{BC} & 0 & 0 \\ 0 & 0 & 1 - q_{AB} & 1 - q_{BC} \\ 0 & 0 & q_{AB} & q_{BC} \end{pmatrix} \quad (3.22)$$

$$\mathbf{p}(t) = \begin{pmatrix} \frac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_B(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_A(t) \cdot n_{BC}(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{AB}(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \quad (3.23)$$

The equation (3.5) is numerically solved for different initial conditions and defect probabilities, in order to compare the dynamic behavior of parts (and defective parts) in  $((AB)C)$ ,  $(A(BC))$  and  $\{ABC\}$ . Figures 3-31, 3-32 and 3-33, show the solution with the initial condition  $\mathbf{n}_0 = (10, 10, 10)$  and  $\mathbf{q} = (0.0, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0)$  and  $\mathbf{q} = (0.0, 0.2)$ , respectively. The yield of  $\{ABC\}$  is slightly better than  $((AB)C)$  and  $(A(BC))$  in all of the three cases, which matches the results obtained by the GA search<sup>17</sup> shown in Figure 3-28. For  $\mathbf{q} = (0.0, 0.0)$ , there is no difference between the solution of  $((AB)C)$  and  $(A(BC))$ . It is observed, however, that the higher defect probability between  $AB$  ( $\mathbf{q} = (0.2, 0.0)$ ) slightly favors the subassembly sequence  $((AB)C)$  (Figure 3-32), while  $\mathbf{q} = (0.0, 0.2)$  slightly favors  $(A(BC))$  (Figure 3-33).

<sup>17</sup>Recall the robot bin-picking simulation used in the GA search terminates at 700 iterations.

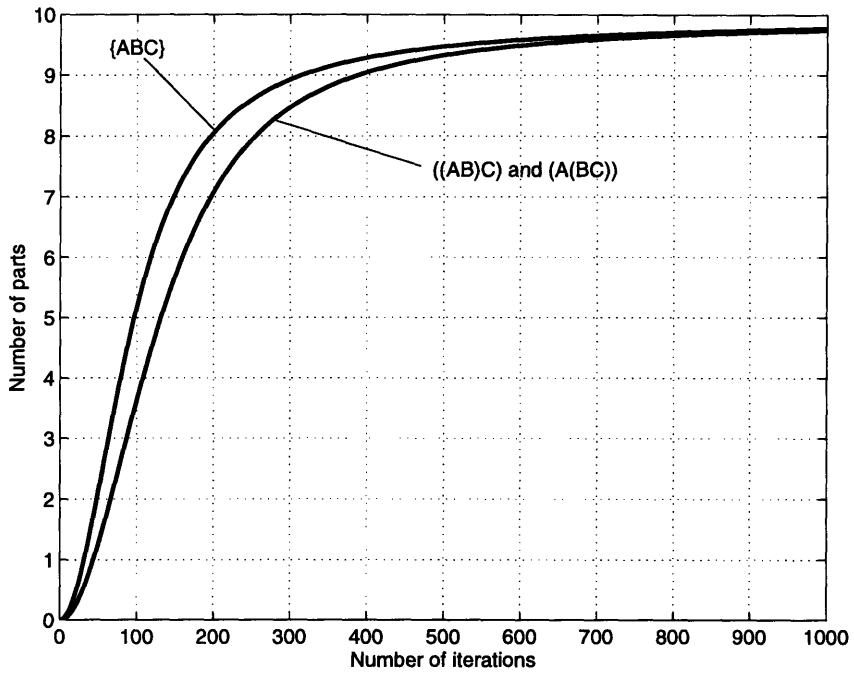


Figure 3-31: Solution with  $n_0 = (10, 10, 10)$  and  $q = (0.0, 0.0)$ .

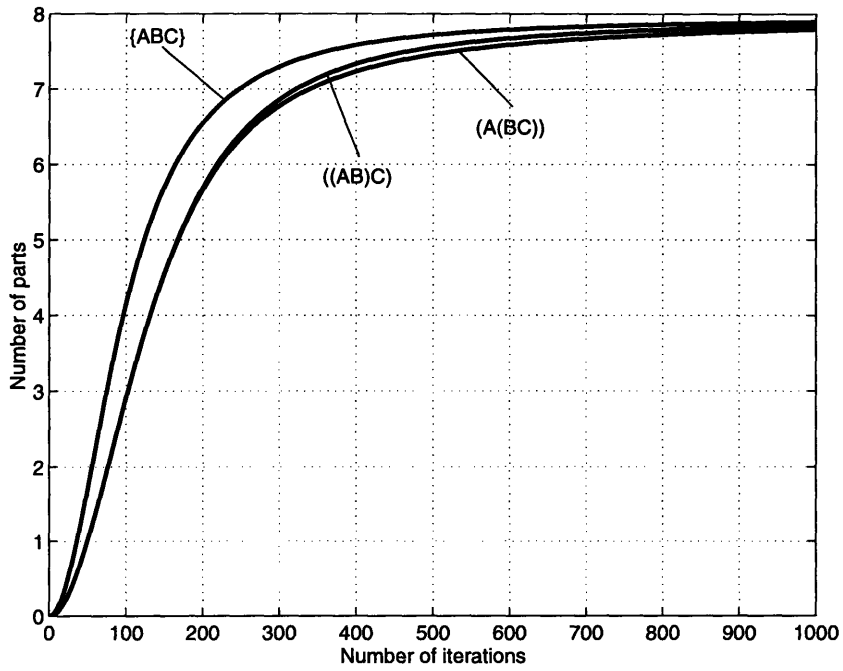


Figure 3-32: Solution with  $n_0 = (10, 10, 10)$  and  $q = (0.2, 0.0)$ .



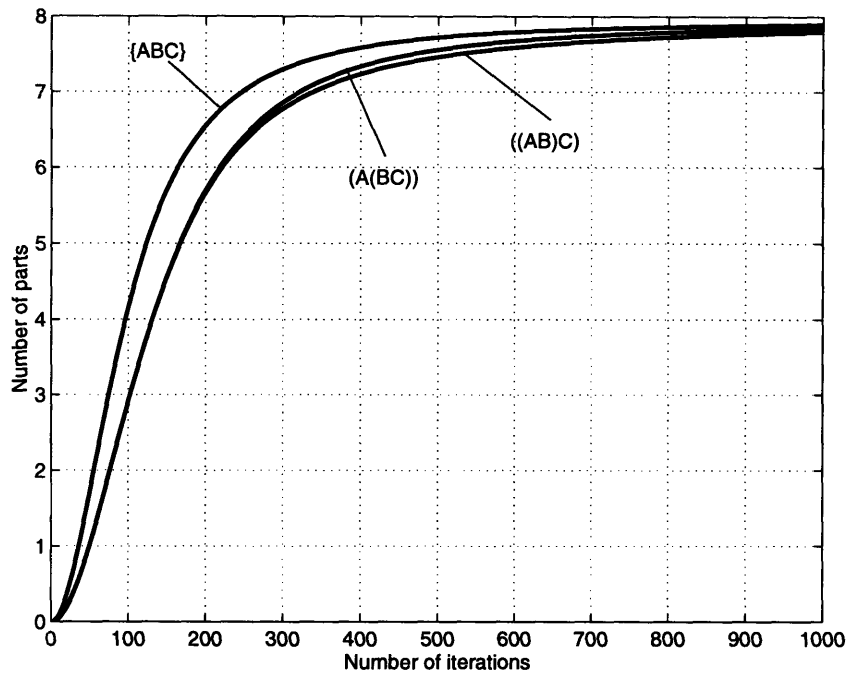


Figure 3-33: Solution with  $\mathbf{n}_0 = (10, 10, 10)$  and  $\mathbf{q} = (0.0, 0.2)$ .

These trends become more prominent with  $\mathbf{n}_0 = (10, 20, 10)$ , as shown in Figures 3-34, 3-35 and 3-36. The result of  $((AB)C)$  and  $(A(BC))$  are identical for  $\mathbf{q} = (0.0, 0.0)$ , which are better than the yield of  $\{ABC\}$ . For  $\mathbf{q} = (0.2, 0.0)$ , the yield of  $((AB)C)$  is approximately 5% better than the yield of  $(A(BC))$ , whereas for  $\mathbf{q} = (0.0, 0.2)$ , the yield of  $(A(BC))$  is approximately 5% better than the yield of  $((AB)C)$ . The above results support the results by GA runs shown in Figure 3-29. It should be noted that in the case of  $\mathbf{n}_0 = (10, 20, 10)$ , the yield of  $\{ABC\}$  goes down to about 50% of the maximum possible yield. This rather counter-intuitive drop of the yield is due to the large number of the middle part  $B$ , which produces a large number of  $AB$ 's and  $BC$ 's in the early stage of iterations. The 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 subassemblies  $AB$  and  $BC$ . By enforcing the subassembly sequence  $((AB)C)$  or  $(A(BC))$ , this excess production of  $AB$  and  $BC$  can be avoided.

Figures 3-37, 3-38 and 3-39 show the solution of equation (3.5) with  $\mathbf{n}_0 = (20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0)$  and  $\mathbf{q} = (0.0, 0.2)$ , respectively. In all cases,  $\{ABC\}$  has the highest rate of the desired assembly during the early iterations. However,  $((AB)C)$

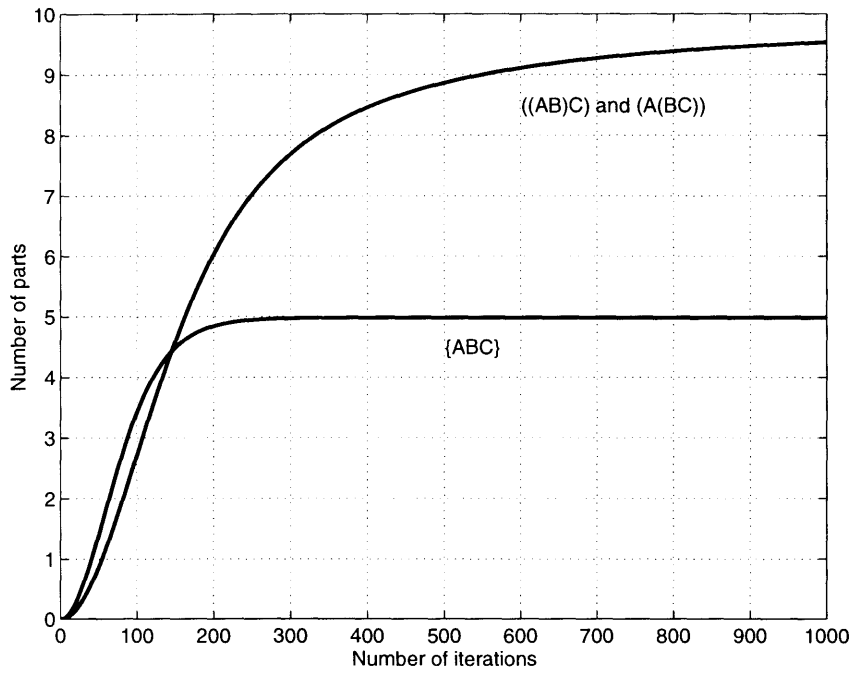


Figure 3-34: Solution with  $\mathbf{n}_0 = (10, 20, 10)$  and  $\mathbf{q} = (0.0, 0.0)$ .

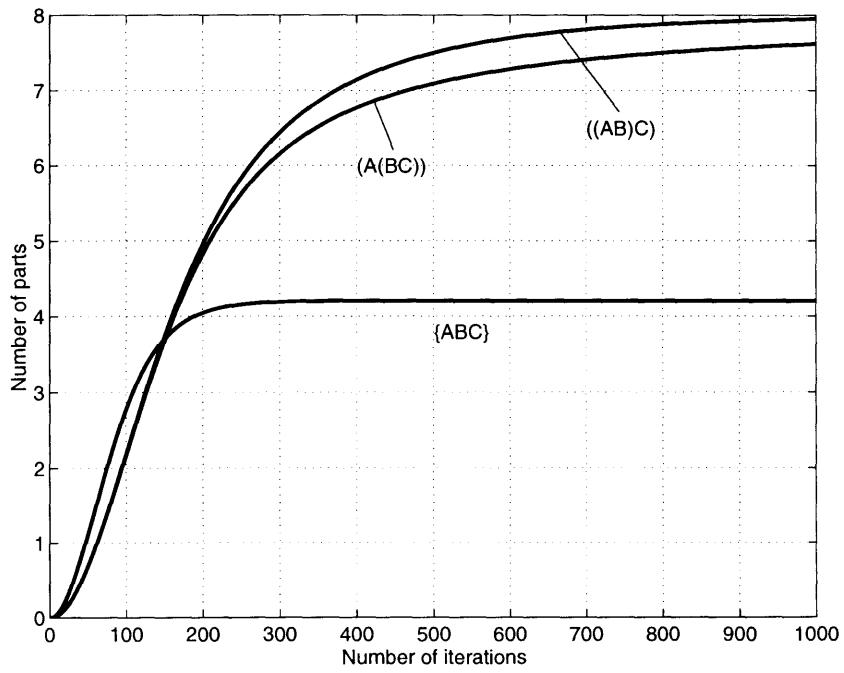


Figure 3-35: Solution with  $\mathbf{n}_0 = (10, 20, 10)$  and  $\mathbf{q} = (0.2, 0.0)$ .

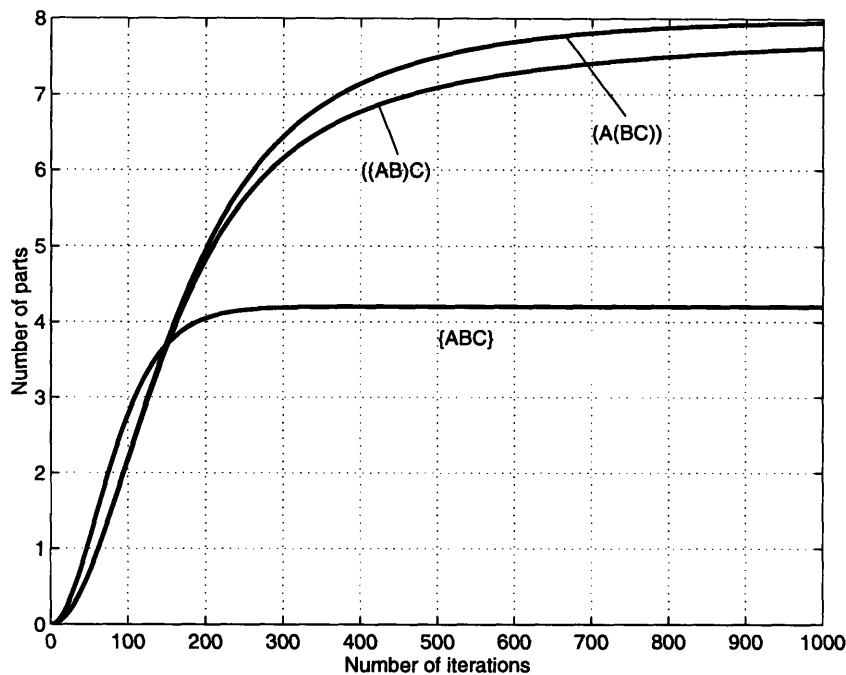


Figure 3-36: Solution with  $\mathbf{n}_0 = (10, 20, 10)$  and  $\mathbf{q} = (0.0, 0.2)$ .

becomes better approximately after the 250-th iteration and scores the best overall yield. Differences in the final yield between  $((AB)C)$  and  $\{ABC\}$  are the largest for  $\mathbf{q} = (0.2, 0.0)$  and the smallest for  $\mathbf{q} = (0.0, 0.2)$ . In particular, in the case of  $\mathbf{q} = (0.0, 0.2)$ , the overall yield is almost identical for  $((AB)C)$ ,  $(A(BC))$  and  $\{ABC\}$ . The GA search in Figure 3-30 found the optimal solutions (*i.e.* conformational switch designs that encode the optimal subassembly sequence) for  $\mathbf{q} = (0.0, 0.0)$  and  $\mathbf{q} = (0.2, 0.0)$ , and found a suboptimal solution within 1% of the optimal solution for  $\mathbf{q} = (0.0, 0.2)$ .

### 3.5.5 Four part self-assembly

The second example is a four part randomized assembly. The initial bin contains a random mixture of four types of parts, part *A*, part *B*, part *C* and part *D*, and the design objective is to maximize the yield of the assembly *ABCD*. The number of *ABCD*'s in the bin is counted after 1400 iterations of Steps 1-3 in Section 3.1. At each evaluation of a chromosome, an average is taken for the count of *ABCD*'s over 50 such runs. The GA runs described in this section have population size of 600 and the number of generations is 900. In the following

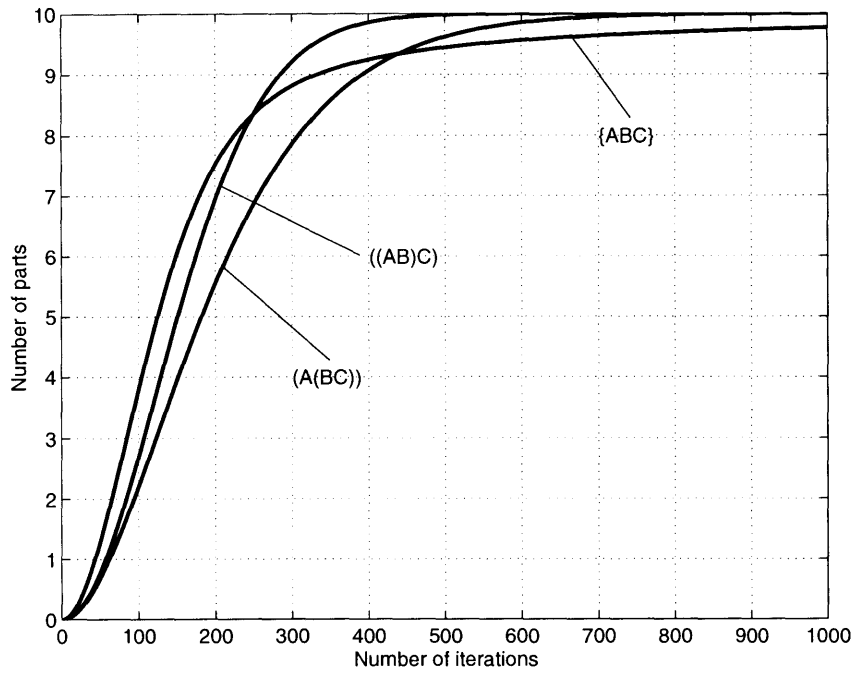


Figure 3-37: Solution with  $\mathbf{n}_0 = (20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.0)$ .

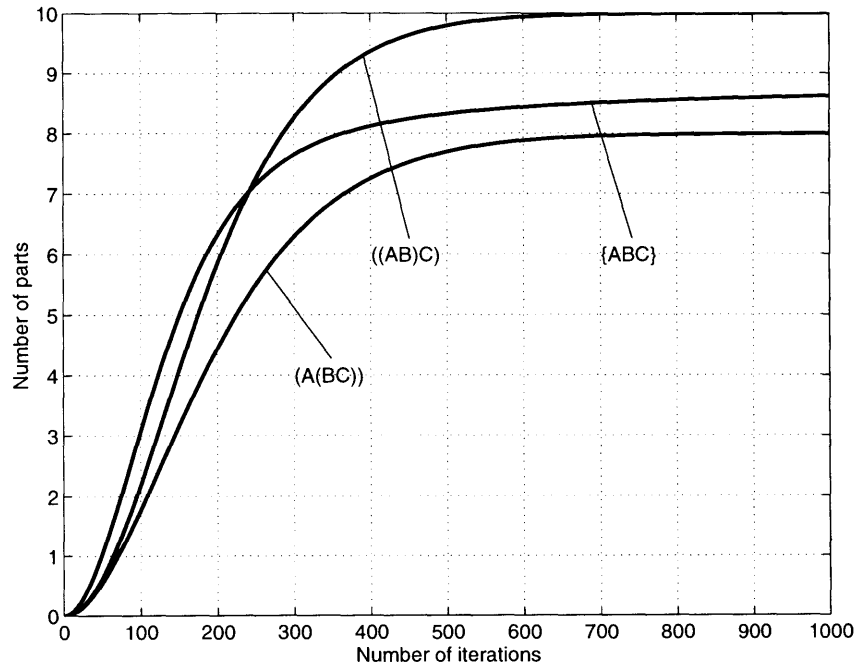


Figure 3-38: Solution with  $\mathbf{n}_0 = (20, 20, 10)$  and  $\mathbf{q} = (0.2, 0.0)$ .

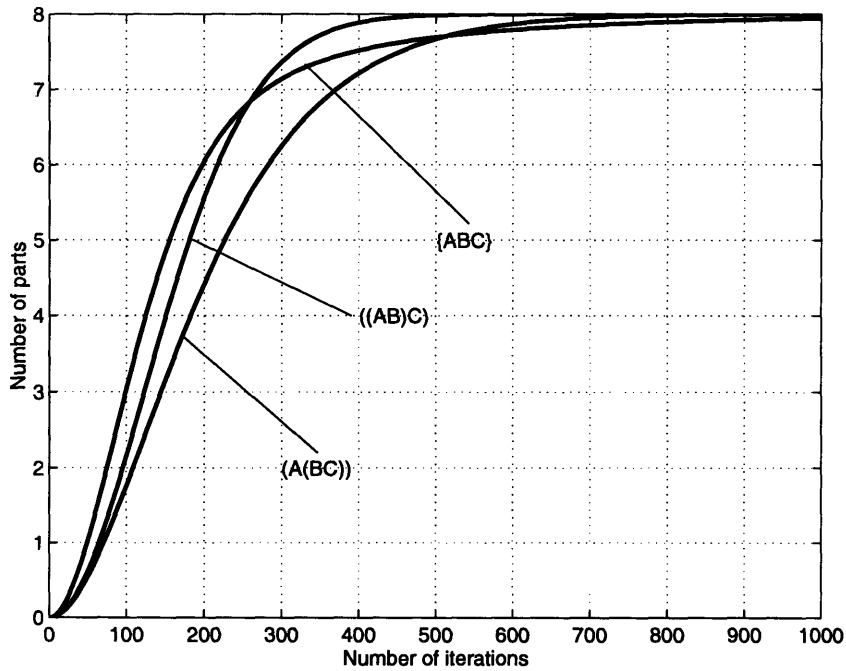


Figure 3-39: Solution with  $\mathbf{n}_0 = (20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.2)$ .

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. Figure 3-40 shows the five non-ambiguous subassembly sequences possible for the four part assembly.

The best designs found by the GA are shown in Figure 3-41 with  $\mathbf{n}_0 = (10, 10, 10, 10)$  and four different defect probabilities: (a)  $\mathbf{q} = (0.0, 0.0, 0.0)$ , (b)  $\mathbf{q} = (0.2, 0.0, 0.0)$ , (c)  $\mathbf{q} = (0.0, 0.2, 0.0)$  and (d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ . As in the case of the three part assembly, the parts are evolved such that *only*  $ABCD$  can form through random mating. The first three results (a), (b) and (c) specify no fixed subassembly sequences. In other words, the parts can be assembled in *any* of the five subassembly sequences shown in Figure 3-40, *i.e.* the design encodes  $\{ABCD\}$ . A fixed subassembly sequence  $((AB)(CD))$  emerged for (d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ , which is realized by conformational changes of part  $B$  and part  $C$  after forming subassemblies  $AB$  and  $CD$ :



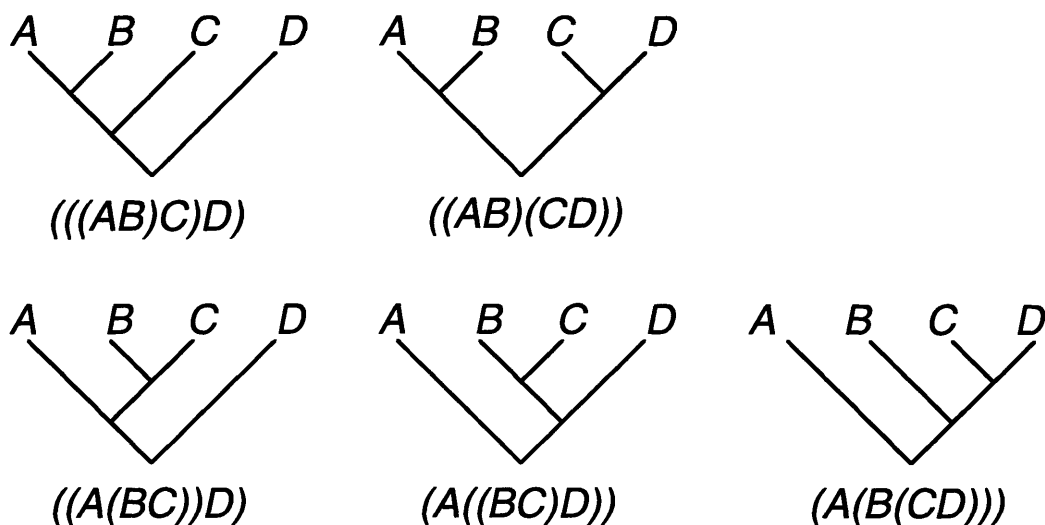


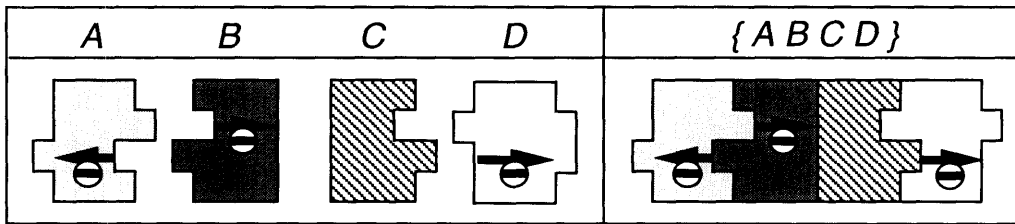
Figure 3-40: Five non-ambiguous subassembly sequences of a four part assembly.



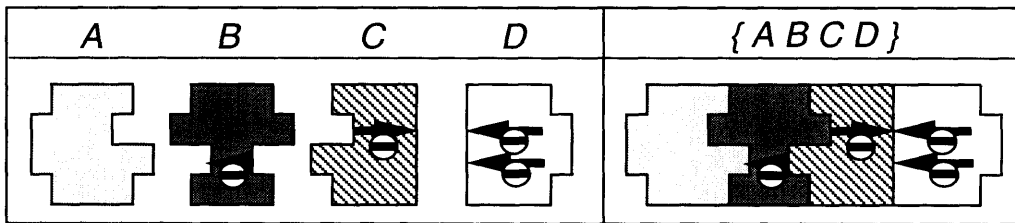
Figure 3-42 shows the results with  $\mathbf{n}_0 = (10, 20, 10, 10)$ . For (a)  $\mathbf{q} = (0.0, 0.0, 0.0)$  and (b)  $\mathbf{q} = (0.2, 0.0, 0.0)$ , a conformational link in part  $B$  causes a  $B-C$  bond to be made only *after* an  $A-B$  bond forms. The final assembly  $ABCD$ , therefore, is built in the order either  $(((AB)C)D)$  or  $((AB)(CD))$ , hence the design encodes the subassembly sequence  $\{(AB)CD\}$ . In the case of (c)  $\mathbf{q} = (0.0, 0.2, 0.0)$ , on the other hand, a conformational link in part  $B$  causes a  $B-C$  bond to be made *before* an  $A-B$  bond forms. Therefore, the design encodes the subassembly sequences  $((A(BC))D)$ ,  $(A((BC)D))$  or  $(A(B(CD)))$ . I refer to the set of these three subassembly sequences as  $\overline{\{(AB)CD\}}$ , since they are the subassembly sequences that are *not* represented by  $\{(AB)CD\}$  among the five possible non-ambiguous subassembly sequences in Figure 3-40. As in the case of  $\mathbf{n}_0 = (10, 10, 10, 10)$ , the resulting design specifies a fixed subassembly sequence  $((AB)(CD))$  for (d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ .

The sequence  $((AB)(CD))$  also emerged for  $\mathbf{n}_0 = (10, 20, 20, 10)$ , with (a)  $\mathbf{q} = (0.0, 0.0, 0.0)$ , (b)  $\mathbf{q} = (0.2, 0.0, 0.0)$  and (d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ , as shown in Figure 3-43. The design with (c)  $\mathbf{q} = (0.0, 0.2, 0.0)$  encodes the subassembly sequence  $(A(B(CD)))$ , which takes the following three-step reactions:

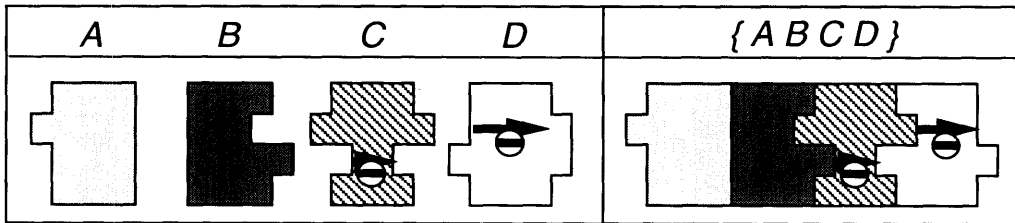
(a)  $\mathbf{q} = (0.0, 0.0, 0.0)$ ; fitness = 10.00;  $n = 32$



(b)  $\mathbf{q} = (0.2, 0.0, 0.0)$ ; fitness = 8.34;  $n = 2$



(c)  $\mathbf{q} = (0.0, 0.2, 0.0)$ ; fitness = 8.42;  $n = 2$



(d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ ; fitness = 7.63;  $n = 3$

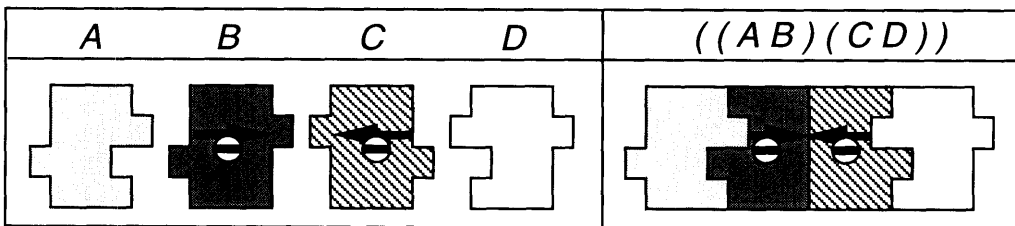
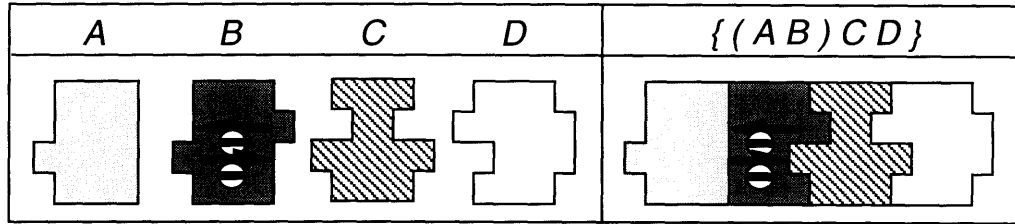
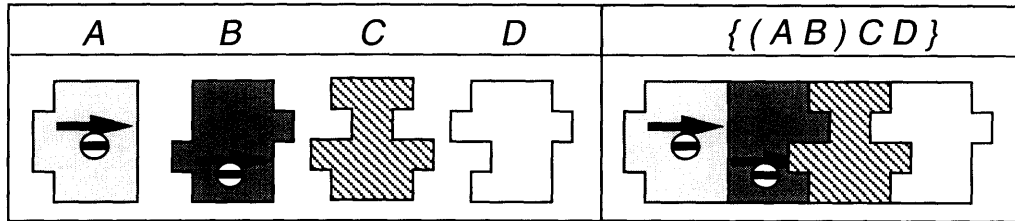


Figure 3-41: Best designs with  $\mathbf{n}_0 = (10, 10, 10, 10)$ .

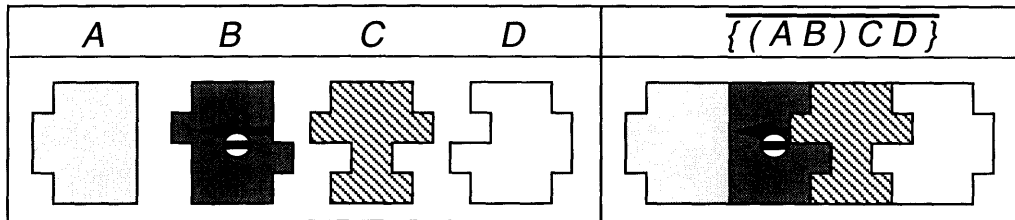
(a)  $\mathbf{q} = (0.0, 0.0, 0.0)$ ; fitness = 9.84;  $n = 2$



(b)  $\mathbf{q} = (0.2, 0.0, 0.0)$ ; fitness = 8.16;  $n = 1$



(c)  $\mathbf{q} = (0.0, 0.2, 0.0)$ ; fitness = 8.28;  $n = 2$



(d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ ; fitness = 8.32;  $n = 1$

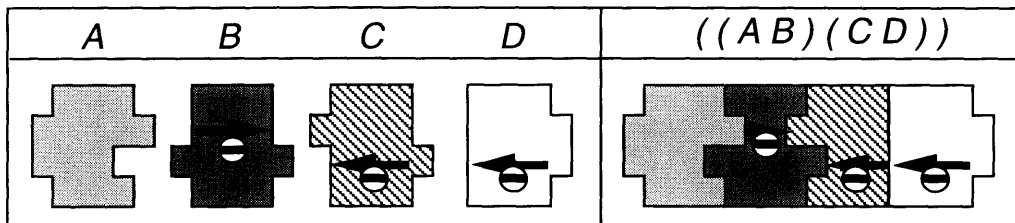
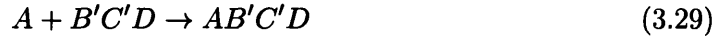


Figure 3-42: Best designs with  $\mathbf{n}_0 = (10, 20, 10, 10)$ .



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 }	$\overline{\{(AB) CD\}}$	(A(B(CD)))
(0.2,0.0,0.2)	((AB) (CD))	((AB) (CD))	((AB) (CD))

Table 3.2: Summary of the results: four part self-assembly.



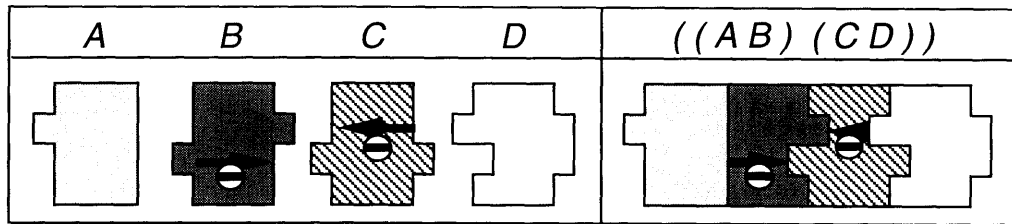
Note in the above results (of Figures 3-41(d), 3-42(d), 3-43(a), (b), (c) and (d)), that *exactly* two conformational links, one in part *B* and one in part *C*, are actually used to encode the two non-ambiguous subassembly sequences ((*AB*)(*CD*)) and (*A*(*B*(*CD*))). Other links are *non-functional* (do not cause conformational changes of a part) or *redundant* (cause conformational changes that do not affect assembly sequences). Similarly, as shown in Figures 3-42(a), (b) and (c), only one conformational link in part *B* is required to encode {(*AB*)*CD*} and  $\overline{\{(AB)CD\}}$ , and no conformational link is required to encode {*ABCD*} (see Figures 3-41(a), (b) and (c))<sup>18</sup>. The summary of these twelve GA runs are shown in Figure 3.2.

### 3.5.6 Rate equation analyses of four part self-assembly

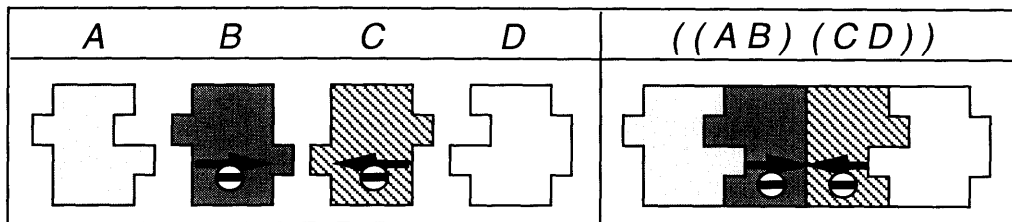
Rate equations (3.5) of four-part self-assembly are formulated in a similar way to the three-part case in Section 3.5.4. The yield of the final assembly *ABCD* is then compared for *all*

<sup>18</sup>Conformational changes of part *D* in Figures 3-41(a) and (c) do not affect assembly sequences, therefore the corresponding links are redundant.

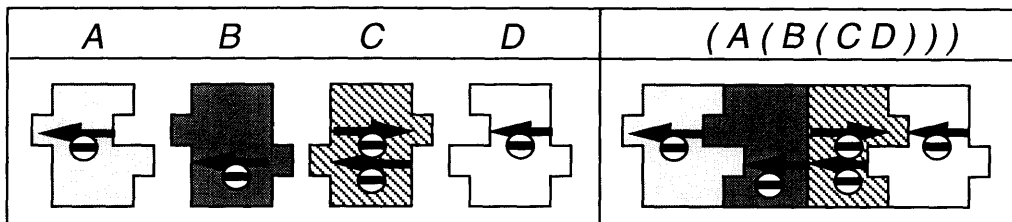
(a)  $\mathbf{q} = (0.0, 0.0, 0.0)$ ; fitness = 9.48;  $n = 1$



(b)  $\mathbf{q} = (0.2, 0.0, 0.0)$ ; fitness = 8.08;  $n = 2$



(c)  $\mathbf{q} = (0.0, 0.2, 0.0)$ ; fitness = 7.98;  $n = 1$



(d)  $\mathbf{q} = (0.2, 0.0, 0.2)$ ; fitness = 7.24;  $n = 1$

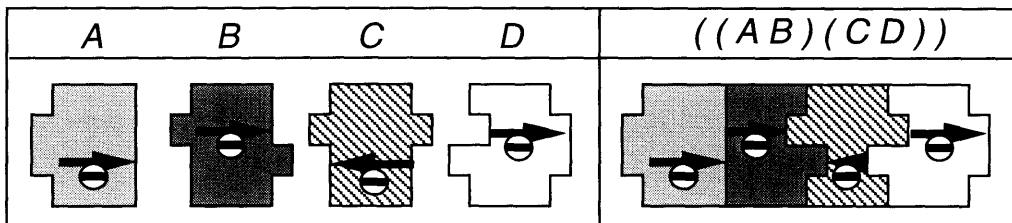


Figure 3-43: Best designs with  $\mathbf{n}_0 = (10, 20, 20, 10)$ .

the subassembly sequences which can be encoded by the conformational switches with two minus devices and two-digit bonding sites, described in Section 3.2.2. Such subassembly sequences can be enumerated by listing combinations of *functional* and *non-redundant* conformational links in the parts, *i.e.* by listing all combinations of conformational links which are necessary and sufficient to encode a set of non-ambiguous subassembly sequences. Since the desired assembly is  $ABCD$ , any conformational links in part  $A$  and part  $D$  are non-functional/redundant. Also, one of any two conformational links in a part pointing the same direction is non-functional/redundant since it is not possible to induce conformational changes via two such conformational links (see Section 3.2.2). Even when two conformational links in a part are pointing in the opposite directions, one of them is still non-functional/redundant since a bond cannot be destroyed once it is formed.

The above discussion leaves us only eight non-redundant combinations of conformational links, which encode five ambiguous subassembly sequences  $\{ABCD\}$ ,  $\{(AB)CD\}$ ,  $\{\overline{(AB)CD}\}$ ,  $\{AB(CD)\}$  and  $\{\overline{AB(CD)}\}$ , and three non-ambiguous subassembly sequences  $\{((AB)C)D\}$ ,  $\{(AB)(CD)\}$  and  $\{A(B(CD))\}$ , as illustrated in Figure 3-44<sup>19</sup>. It is shown, therefore, that the conformational switch model *cannot* encode two of the non-ambiguous subassembly sequences  $\{(A(BC))D\}$  and  $\{A((BC)D)\}$  in Figure 3-40. This is due to the fact that the conformational switch model does not allow propagation of conformational change through parts<sup>20</sup>.

The rate equations for each of the eight subassembly sequences in Figure 3-45 are formulated and solved numerically to compare the yield of the final assembly  $ABCD$ . The results are obtained with three different initial conditions:  $\mathbf{n}_0 = (10, 10, 10, 10)$ ,  $\mathbf{n}_0 = (10, 20, 10, 10)$  and  $\mathbf{n}_0 = (10, 20, 20, 10)$ . For each of the three initial conditions, four different defect probabilities are tried:  $\mathbf{q} = (0.0, 0.0, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0, 0.0)$ ,  $\mathbf{q} = (0.0, 0.2, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0, 0.2)$ . These  $3 \times 4 = 12$  conditions correspond to the conditions of GA runs shown in Figures 3-41, 3-42 and 3-43. Figures 3-45–3-48 show the solution of the equation 3.5 with  $\mathbf{n}_0 = (10, 10, 10, 10)$  and with  $\mathbf{q} = (0.0, 0.0, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0, 0.0)$ ,  $\mathbf{q} = (0.0, 0.2, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0, 0.2)$ , respectively. For the first three cases, the sequence  $\{ABCD\}$  scores the best at 1400 iterations, whereas it is outperformed by  $\{(AB)(CD)\}$  after approximately 500

<sup>19</sup>Some extensions are necessary to the conformational switch model, in order to encode  $\{A(BC)D\}$ . This issue is discussed in Section 3.5.8.

<sup>20</sup>An extension of the conformational switch model which can encode  $\{(A(BC))D\}$  and  $\{A((BC)D)\}$  is discussed in Section 3.5.8.

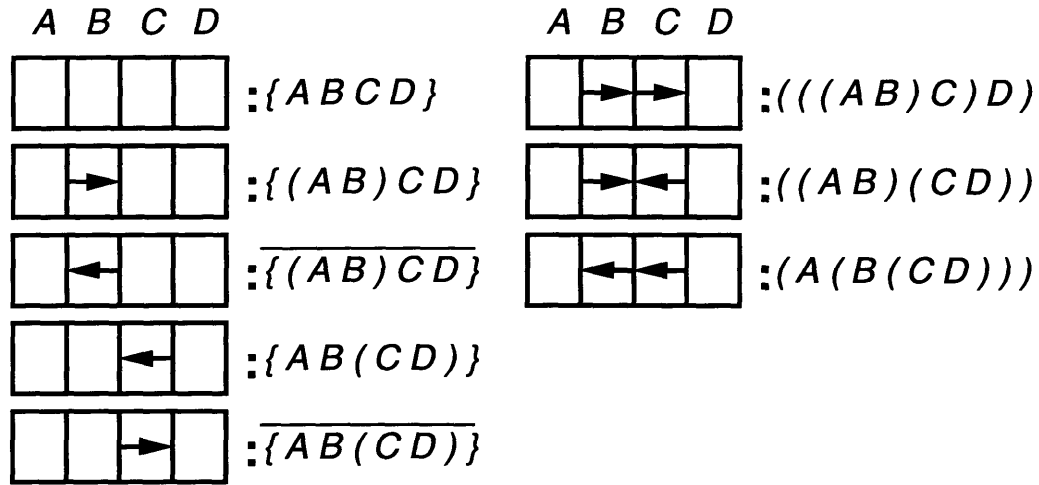


Figure 3-44: Eight possible subassembly sequences.

iterations for  $\mathbf{q} = (0.2, 0.0, 0.2)^{21}$ . In all cases, the results are consistent with the ones by GA shown in Figure 3-41.

The solutions with  $\mathbf{n}_0 = (10, 20, 10, 10)$  are shown in Figures 3-49–3-52. In this case, the yields of the sequences  $\{ABCD\}$ ,  $\{AB(CD)\}$  and  $\overline{\{AB(CD)\}}$  drop to approximately 50% of the maximum possible yield. This situation is similar to the case of the sequence  $\{ABC\}$  with  $\mathbf{n}_0 = (10, 20, 10)$  in the three part assembly, where excess production of  $AB$  and  $BC$  at the early stage of iteration causes the shortage of  $A$ 's and  $B$ 's later on. The sequences which do not specify the assembly order of  $A$ ,  $B$  and  $C$  (or  $CD$ ) perform poorly due to excess production of intermediate subassemblies such as  $AB$ ,  $BC$  or  $BCD$ . As a consequence, the sequences  $\{(AB)CD\}$  and  $\overline{\{(AB)CD\}}$  yield the best with  $\mathbf{q} = (0.0, 0.0, 0.0)$  (Figure 3-49), the sequence  $((AB)(CD))$  is the best with  $\mathbf{q} = (0.2, 0.0, 0.0)$  (Figure 3-50) and  $\mathbf{q} = (0.2, 0.0, 0.2)$  (Figure 3-52), and the sequence  $\overline{\{(AB)CD\}}$  is the best with  $\mathbf{q} = (0.0, 0.2, 0.0)$  (Figure 3-51). The GA found a design that encodes the optimal subassembly sequence for all cases except  $\mathbf{q} = (0.2, 0.0, 0.0)$ , where the design encodes the suboptimal sequence  $\{(AB)CD\}$  within 5% of the yield of the optimal sequence (see Figure 3-42(b)).

Similar drops of yield in some subassembly sequences are observed in the solutions with  $\mathbf{n}_0 = (10, 20, 20, 10)$ . As shown in Figures 3-53–3-56, however, the drop occurs to *all* of the ambiguous subassembly sequences,  $\{ABCD\}$ ,  $\{(AB)CD\}$ ,  $\overline{\{(AB)CD\}}$ ,  $\{AB(CD)\}$

<sup>21</sup>Recall the robot bin-picking simulation used in the GA search terminates at 1400 iterations.

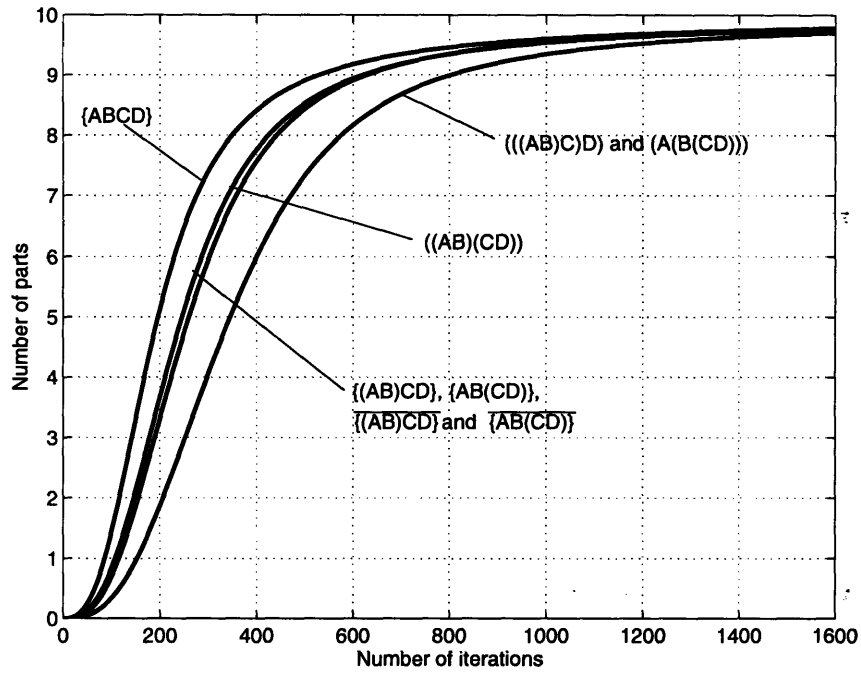


Figure 3-45: Solution with  $n_0 = (10, 10, 10, 10)$  and  $q = (0.0, 0.0, 0.0)$ .

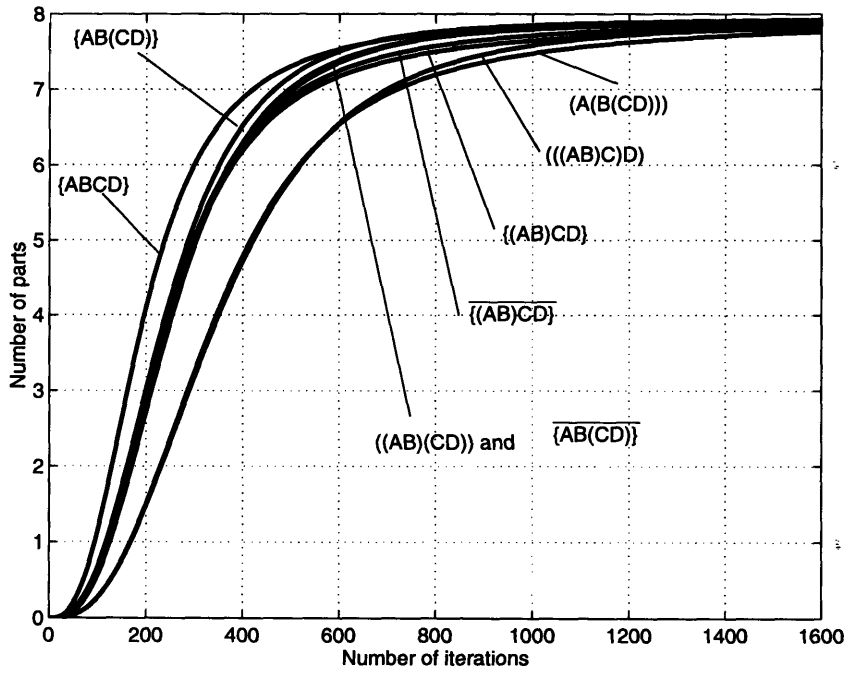


Figure 3-46: Solution with  $n_0 = (10, 10, 10, 10)$  and  $q = (0.2, 0.0, 0.0)$ .

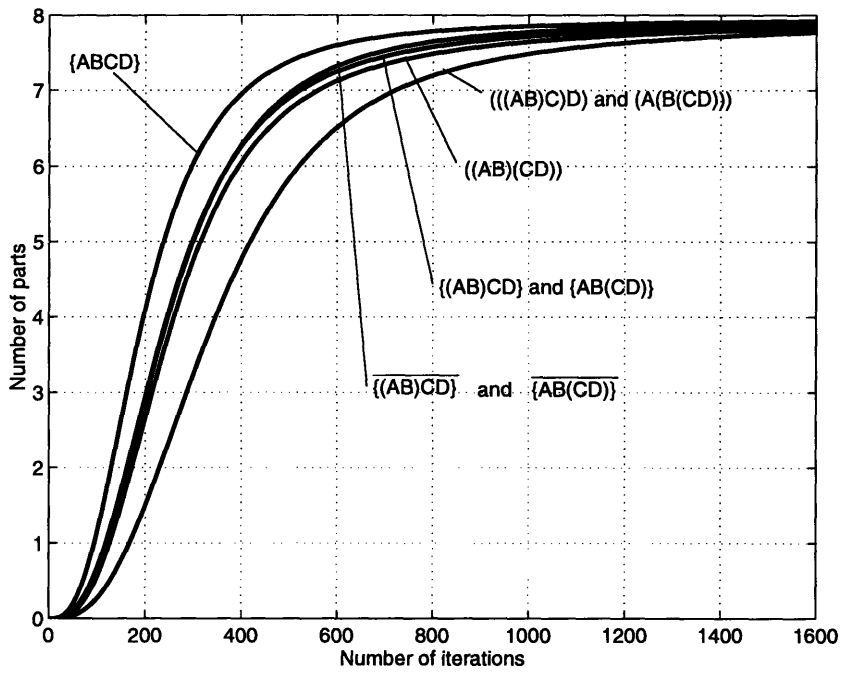


Figure 3-47: Solution with  $n_0 = (10, 10, 10, 10)$  and  $q = (0.0, 0.2, 0.0)$ .

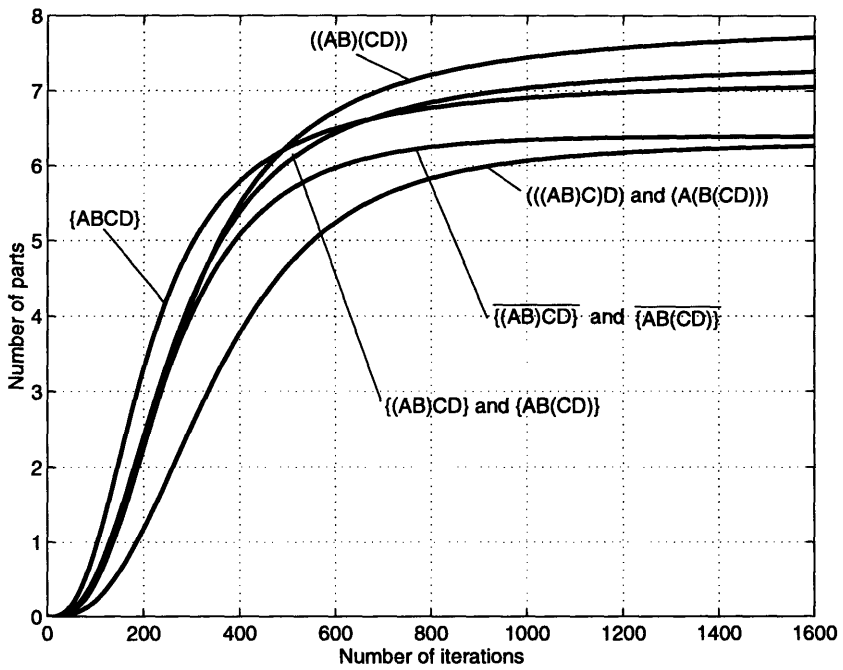


Figure 3-48: Solution with  $n_0 = (10, 10, 10, 10)$  and  $q = (0.2, 0.0, 0.2)$ .

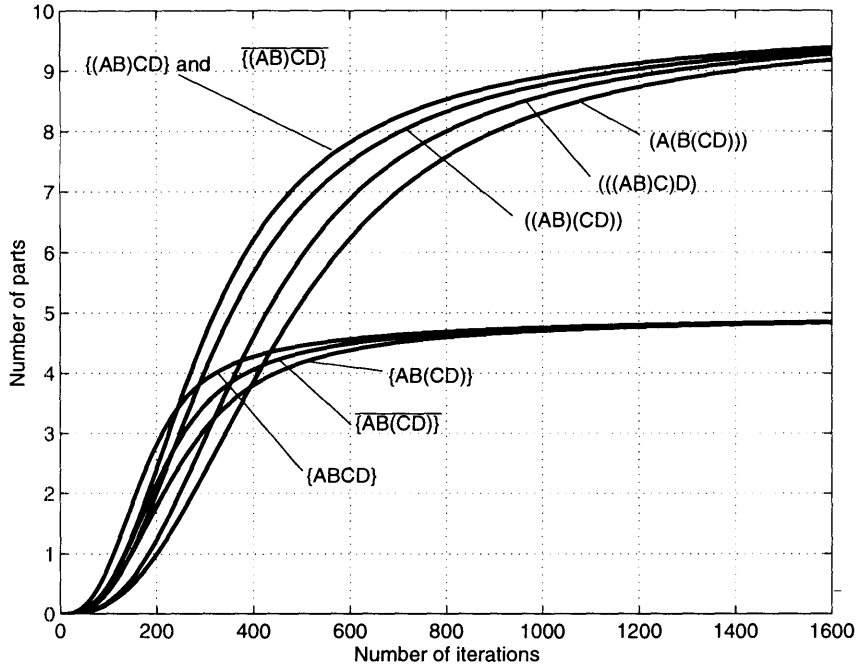


Figure 3-49: Solution with  $\mathbf{n}_0 = (10, 20, 10, 10)$  and  $\mathbf{q} = (0.0, 0.0, 0.0)$ .

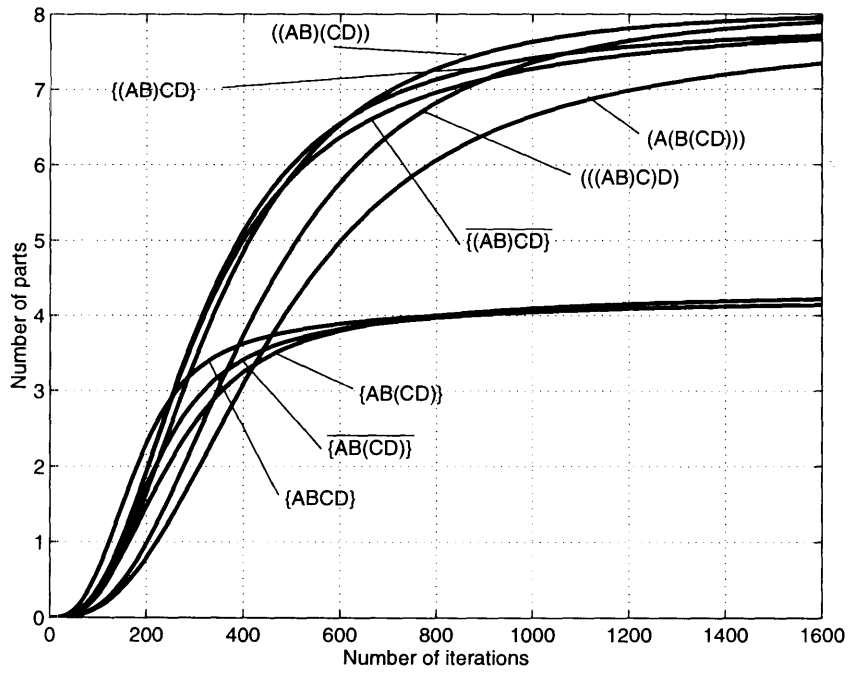


Figure 3-50: Solution with  $\mathbf{n}_0 = (10, 20, 10, 10)$  and  $\mathbf{q} = (0.2, 0.0, 0.0)$ .

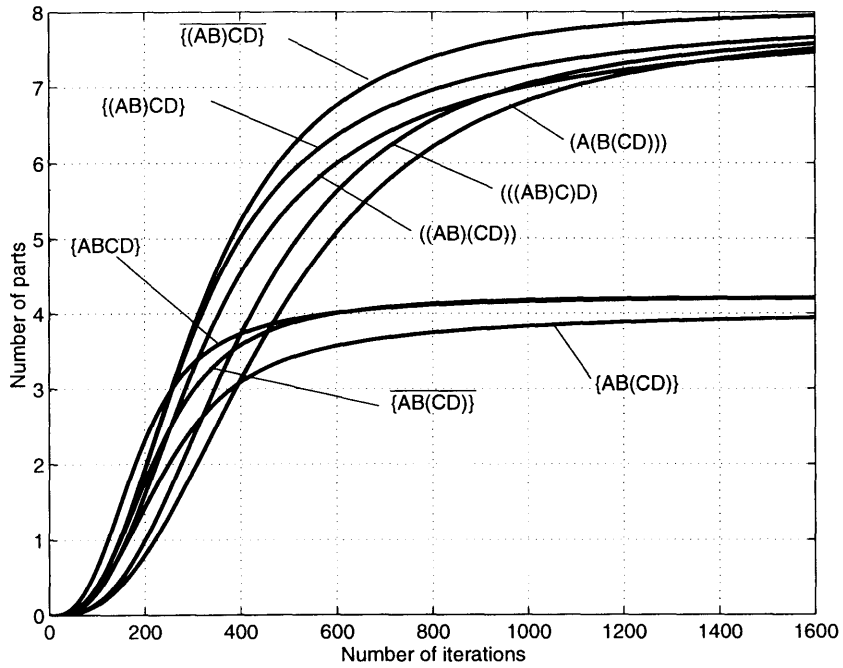


Figure 3-51: Solution with  $\mathbf{n}_0 = (10, 20, 10, 10)$  and  $\mathbf{q} = (0.0, 0.2, 0.0)$ .

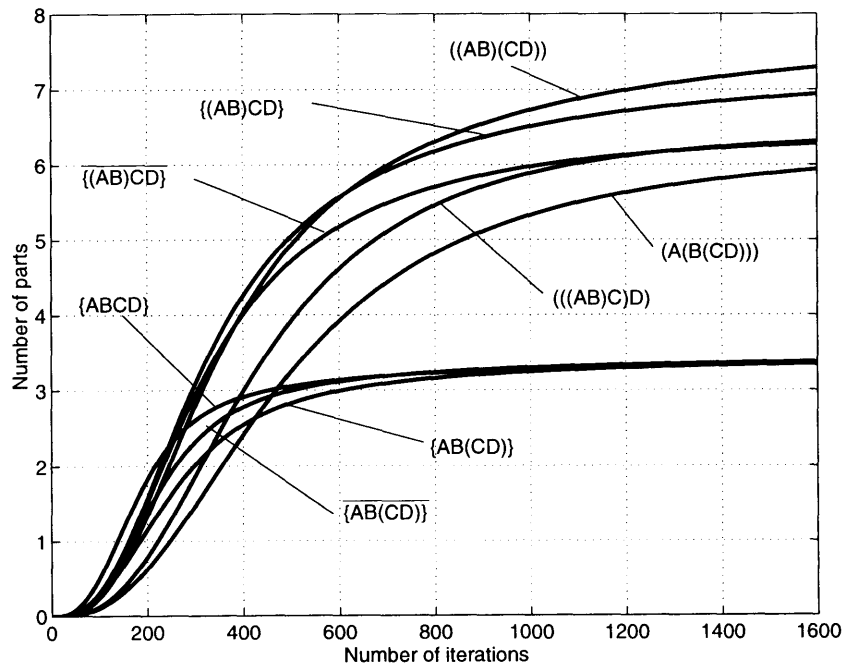


Figure 3-52: Solution with  $\mathbf{n}_0 = (10, 20, 10, 10)$  and  $\mathbf{q} = (0.2, 0.0, 0.2)$ .



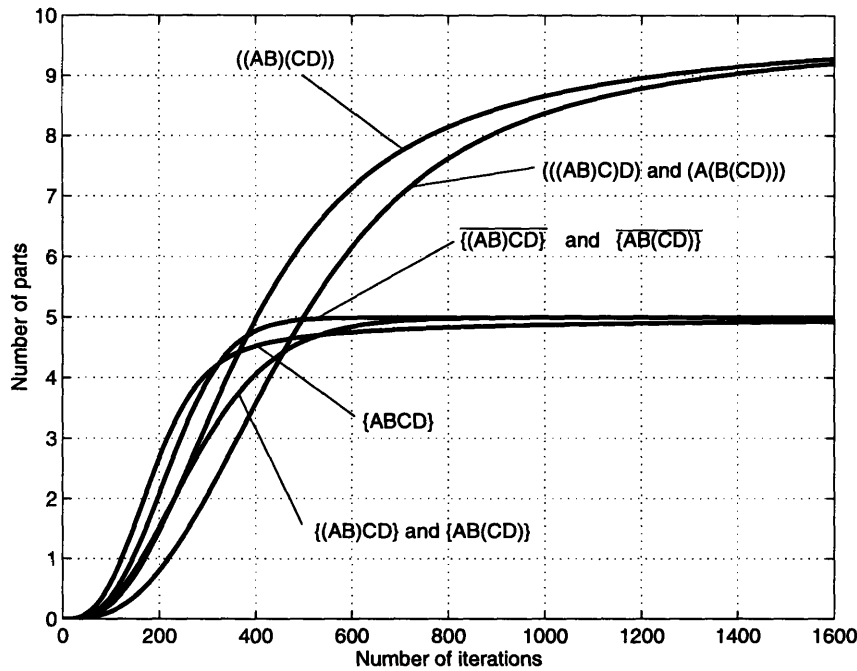


Figure 3-53: Solution with  $\mathbf{n}_0 = (10, 20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.0, 0.0)$ .

and  $\overline{\{AB(CD)\}}$ . The sequence  $((AB)(CD))$  yields the best for  $\mathbf{q} = (0.0, 0.0, 0.0)$ ,  $\mathbf{q} = (0.2, 0.0, 0.0)$ , and  $\mathbf{q} = (0.2, 0.0, 0.2)$ , and the sequences  $(((AB)C)D)$  and  $(A(B(CD)))$  yield the best for  $\mathbf{q} = (0.0, 0.2, 0.0)$ . These results are consistent with the ones found by the GA shown in Figure 3-43.

### 3.5.7 Sensitivity of yield to the initial concentration of parts

The optimal subassembly sequences encoded by conformational switch designs are  $\{ABCD\}$  for  $\mathbf{n}_0 = (10, 10, 10, 10)$  with no defect, *i.e.* no order of assembly is specified as shown in Figure 3-45. On the other hand,  $\{ABCD\}$  yields are very low with different initial concentration of parts as shown in Figures 3-49 and 3-53. This seems to be a general property of ambiguous subassembly sequences: their yield change dramatically depending on the initial part concentration. In fact, the yield of ambiguous subassembly sequences are quite sensitive to the change in the initial concentration of parts. Figure 3-57 shows the solution of rate equations with  $\mathbf{n}_0 = (10, 11, 11, 10)$  and  $\mathbf{q} = (0.0, 0.0, 0.0)$  (compare with Figure 3-45). Even with the slightest change in  $\mathbf{n}_0$ , the yield of ambiguous subassembly sequences

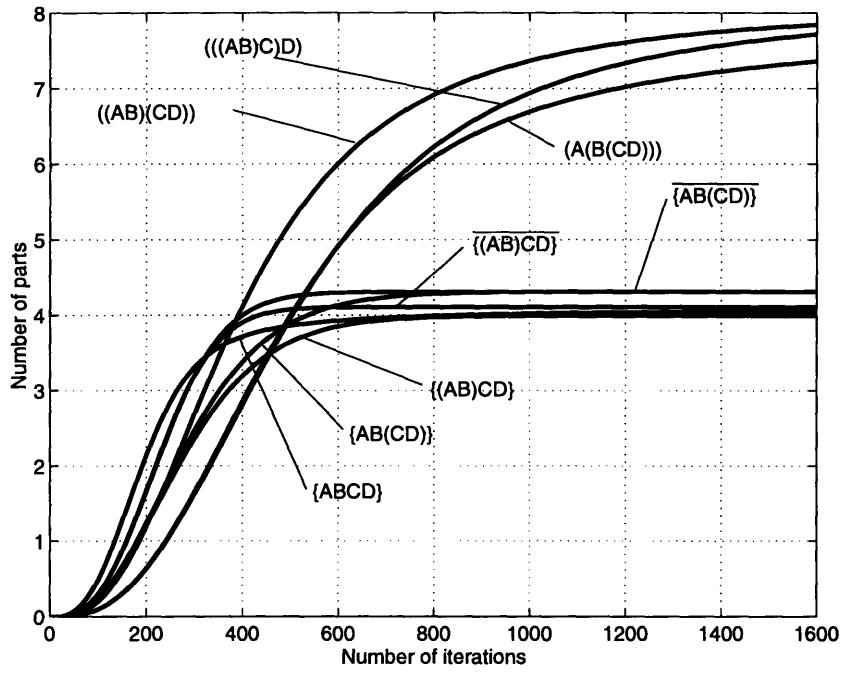


Figure 3-54: Solution with  $n_0 = (10, 20, 20, 10)$  and  $q = (0.2, 0.0, 0.0)$ .

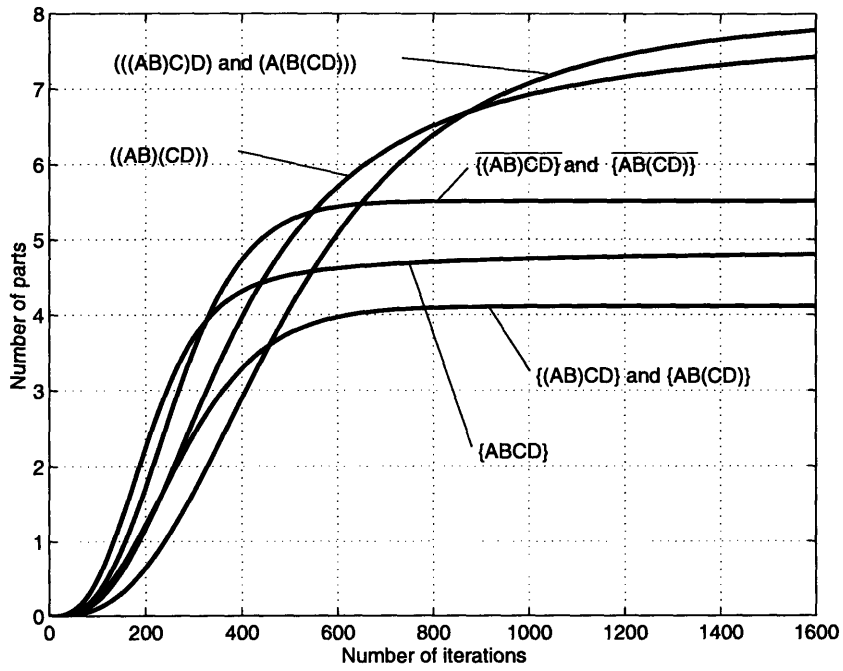


Figure 3-55: Solution with  $n_0 = (10, 20, 20, 10)$  and  $q = (0.0, 0.2, 0.0)$ .

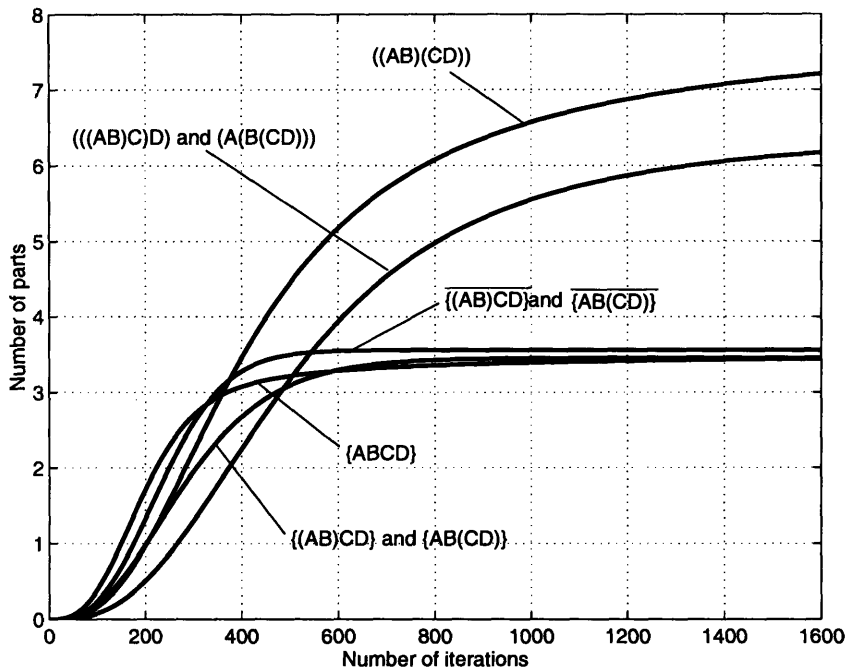


Figure 3-56: Solution with  $\mathbf{n}_0 = (10, 20, 20, 10)$  and  $\mathbf{q} = (0.2, 0.0, 0.2)$ .

are approximately 10% lower than the ones of non-ambiguous subassembly sequences.

On the other hand, the initial part concentration has little effect on the yield of non-ambiguous subassembly sequences. This robustness of non-ambiguous subassembly sequences against the initial part concentration is a great advantage in real-world self-assembly processes, *e.g.* biological self-assembly, where realizing a precise and uniform concentration of parts is extremely difficult.

### 3.5.8 Encoding power of a conformational switch model

One must note the encoding power of a conformational switch model, in order to say that the switch design obtained by a GA actually encodes the optimal subassembly sequence. There could be a subassembly sequence that *cannot* be encoded by a conformational switch design, but that *is* better than any subassembly sequences encoded by the switch design. In fact,  $((A(BC))D)$  and  $(A((BC)D))$ , the subassembly sequences the conformational switch model *cannot* encode, yield better than  $(A(B(CD)))$ , the best sequence obtained by the GA in the four part self-assembly with  $\mathbf{n}_0 = (10, 20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.2, 0.0)$ , as shown

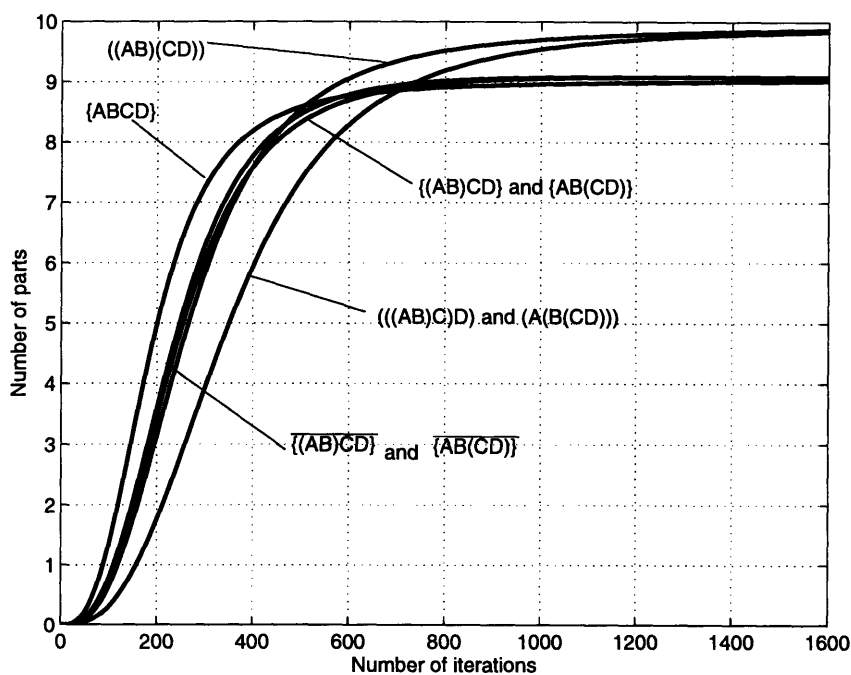


Figure 3-57: Solution with  $\mathbf{n}_0 = (10, 11, 11, 10)$  and  $\mathbf{q} = (0.0, 0.0, 0.0)$ .

in Figure 3-58. It should be emphasized, however, that the sequence found by the GA is the best among the sequences that can be encoded by the conformational switch design.

For comparison, Figure 3-58 also shows the solution of rate equations for three other un-encodable subassembly sequences,  $(\{ABC\}D)$ ,  $(A\{BCD\})$  and  $\{A(BC)D\}$ . The sequences  $(\{ABC\}D)$  and  $(A\{BCD\})$  actually outperform the best encodable sequences  $(((AB)C)D)$  and  $(A(B(CD)))$ . The sequences  $((A(BC))D)$  and  $(A((BC)D))$  also perform better than any encodable sequences in the cases with  $\mathbf{n}_0 = (10, 20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.0, 0.0)$ , and with  $\mathbf{n}_0 = (10, 20, 20, 10)$  and  $\mathbf{q} = (0.2, 0.0, 0.0)$ . The differences in yield, however, are marginal.

Only a few modifications/extensions are necessary in the current conformational switch model to encode the five un-encodable subassembly sequences in Figure 3-58,  $\{A(BC)D\}$ ,  $((A(BC))D)$ ,  $(A((BC)D))$ ,  $(\{ABC\}D)$  and  $(A\{BCD\})$ . In order to encode  $\{A(BC)D\}$ , I only need to modify priority to upstream propagation (Figure 3-11) such that conformational changes can propagate in *both* direction when it is possible<sup>22</sup>. Figure 3-59 illustrates an

<sup>22</sup> Appropriate changes in the definition of unstable bond are also required.

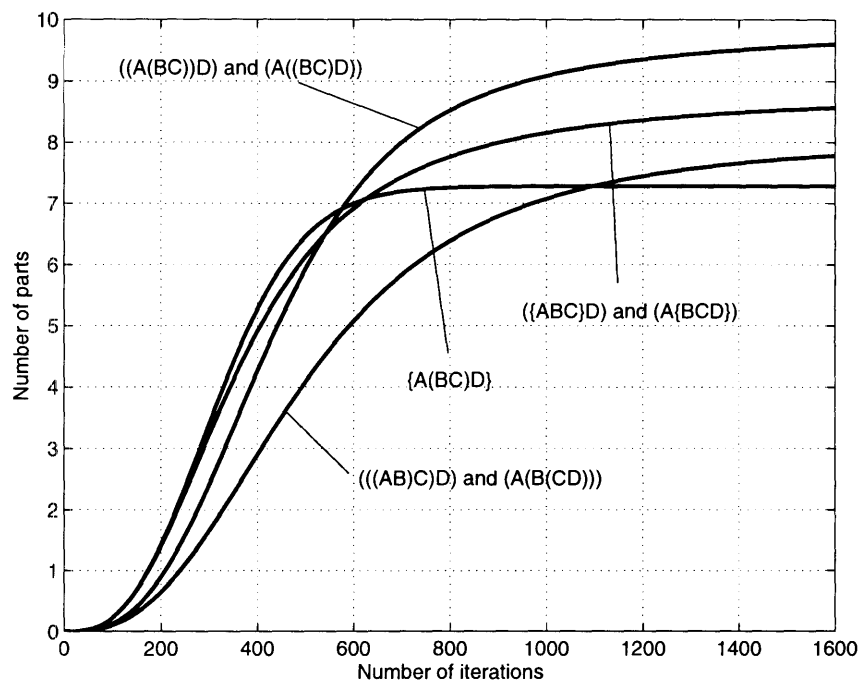


Figure 3-58: Solution with  $\mathbf{n}_0 = (10, 20, 20, 10)$  and  $\mathbf{q} = (0.0, 0.2, 0.0)$ : comparison with un-encodable subassembly sequences

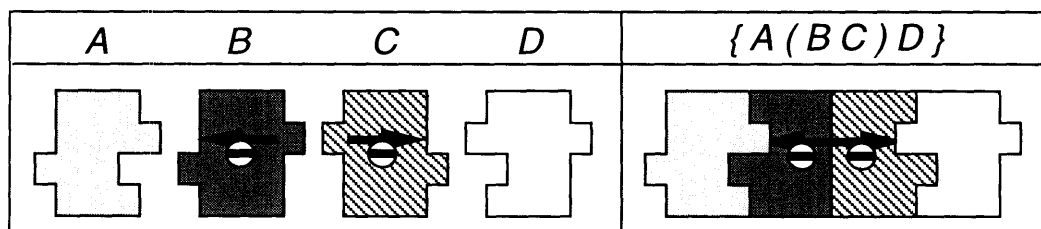


Figure 3-59: Conformational switch design that encodes  $\{A(BC)D\}$

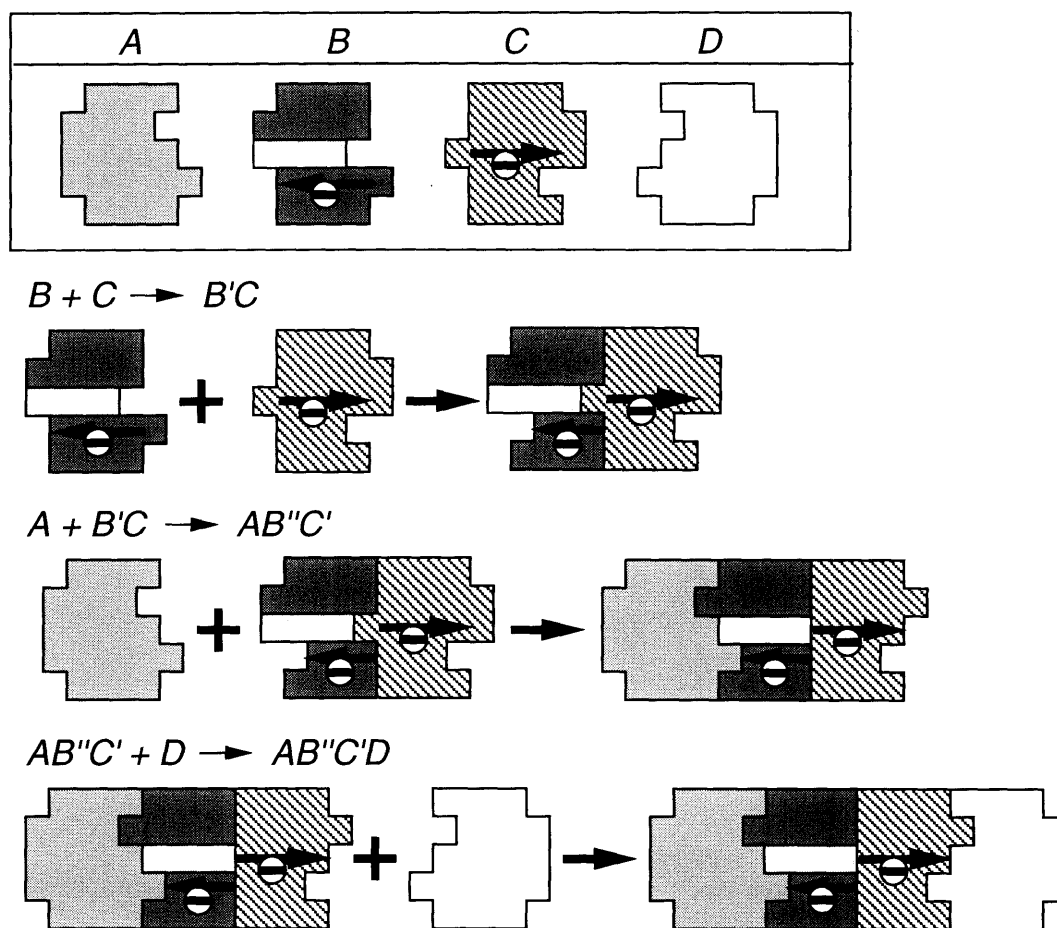


Figure 3-60: Conformational switch design that encodes  $((A(BC))D)$ .

example of a conformational switch design that encodes  $\{A(BC)D\}$  with this modification.

The sequences  $((A(BC))D)$ ,  $(A((BC)D))$ ,  $(\{ABC\}D)$  and  $(A\{BCD\})$  can be encoded by introducing the *sliding bar mechanism* described in [49], which allows propagation of conformational change through multiple parts, and an additional “digit”<sup>23</sup>. The definition of three types of bonding can be defined, for example, as analogous to the two-digit case. Namely, two bond sites form a stable bond if  $\forall i a_i + b_i \leq 0$ , unstable bond if  $\exists_1 j a_j + b_j = 1$  and  $\forall i \neq j a_i + b_i \leq 0$ , and no bond otherwise, where  $i, j \in \{1, 2, 3\}$ . Figure 3-60 illustrates an example of a conformational switch design that encodes  $((A(BC))D)$  with this extension and its three-step reactions.

<sup>23</sup>It seems that there is no two-digit switch designs that encodes these four sequences. Proving/disproving this would be a part of future work.

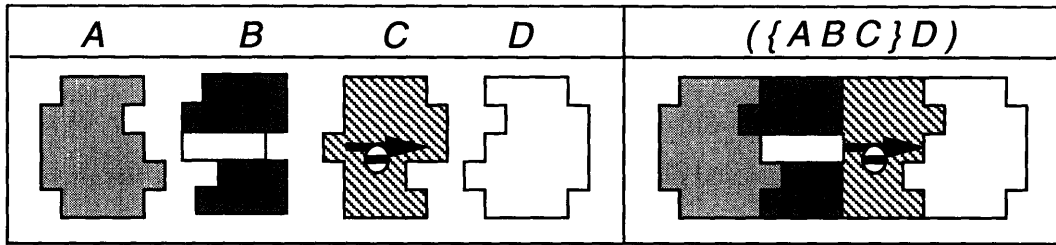


Figure 3-61: Conformational switch design that encodes  $(\{ABC\}D)$ .

Note that the minus link in part  $B$  allows parts  $A$  and  $B$  to bond only *after* parts  $B$  and  $C$  bond. By removing the link, therefore, one can construct a switch design that encodes  $(\{ABC\}D)$ , as shown in Figure 3-61.

Conformational switch designs that encode  $(A((BC)D))$  and  $(A\{BCD\})$  can be obtained by horizontally flipping the designs in Figures 3-60 and 3-61, respectively.

### 3.6 Summary

#### Two part one-dimensional self-assembly with sliding bar mechanisms

Examples in Sections 3.4.3–3.4.6 indicate that sliding bar mechanisms can temporarily change the concentration of parts in the bin by forming temporal assemblies and disassemble them. The optimal switch designs by genetic search, that maximizes the yield of a desired assembly, exhibit this behavior when the part concentration in the initial bin is significantly different from the ones in the desired assembly. Formation of such temporal assemblies can increase the chance of scarce parts being found, by decreasing the overall part count in the bin. If an  $AB$  assembly is desired, for example, and there are many more part  $A$  than part  $B$  in the bin, assemblies containing only part  $A$  (e.g.  $AA$ ), will decrease the overall part count and in turn increase the probability of randomly picking part  $B$ .

## Subassembly generation in multi-part one-dimensional self-assembly with minus devices

As discussed in Sections 3.5.3–3.5.6, minus devices can encode a particular subassembly sequence due to conformational changes of parts during one-dimensional self-assembly. An optimal subassembly sequence that maximizes the yield of a desired assembly can be found via genetic search over a space of parameterized conformational switch designs, rather than a space of subassembly sequences. The resulting switch design encodes the optimal subassembly sequence so that the desired assemblies are put together only in that sequence. The results of genetic search and rate equation analyses reveal that the optimal subassembly sequence depends on the initial concentration of parts and the defect probabilities during self-assembly. More specifically, the results seem to indicate the following “rules of thumb” to design conformational switches for the general  $n$ -part one-dimensional self-assembly:

- Non-ambiguous subassembly sequences yields better than ambiguous subassembly sequences.
- Parallel subassembly sequences yield better than linear subassembly sequences.
- Abundant parts should be assembled earlier rather than later.
- Parts with high defect probability should be assembled earlier rather than later.



---

# Theory of One-dimensional Self-assembling Automata

---

In this chapter, an abstract model of self-assembling systems is presented [52] where assembly instructions are written as local rules that specify conformational changes of a component. The model, the *self-assembling automaton*, is defined as a sequential rule-based machine that operates on one-dimensional strings of symbols. An algorithm is provided for constructing a self-assembling automaton which self-assembles a one-dimensional string of distinct symbols in a given subassembly sequence. Classes of self-assembling automata are then defined based on three classes of subassembly sequences described by assembly grammars. The minimum number of conformations is provided which is necessary to encode instances of each class of subassembly sequences. It is proven that the rules corresponding to the two types of conformational switches described in the previous sections, with three conformations for each component, are enough to encode any subassembly sequences of a string of distinct symbols with arbitrary length.

## 4.1 Motivation

In Section 3.5, it was found that conformational switches can encode one or more subassembly sequences, and the encodable subassembly sequences depend on the conformational switches employed. In particular, as discussed in Section 3.5.8, some subassembly sequences *cannot* be encoded by conformational switches no matter how many conformations we assumed. This raises the following questions:

- Is it possible to tell whether a subassembly sequence can be encoded by given types of conformational switches?
- If so, how many conformations (or switch states) are necessary to encode a given subassembly sequence?

The relationship between subassembly sequences and types of conformational switch is analogous to the one between languages and “machines” (models of computation) in the theory of computation [36], with a subassembly sequence being a language, and conformational switches that encodes the subassembly sequence being a machines that accepts the language. Under this view, the above problems can be seen as analogs of a problem of finding a class of machines that accepts a given language, and a problem of finding the minimum number of states of the machine that accepts a given instance of the language.

This analogy motivated me to address the above problems in the case of general self-assembling systems, not in the case of the particular implementation of conformational switches such as sliding bar mechanisms and minus devices. More specifically, the above problems are approached by developing a formal model of self-assembling systems which abstracts the built-in assembly instructions in the form of conformational switches, and by identifying classes of self-assembling systems based on subassembly sequences in which the components of the systems self-assemble. An abstract representation of such a system is defined as a sequential machine that processes one-dimensional strings of symbols, which I will refer to as an one-dimensional *self-assembling automaton*. Before proceeding to a formal definition of a self-assembling system, two simple examples are discussed which emphasize the essential aspect of machines of this type, by illustrating how conformational switches can encode a subassembly sequence.

## 4.2 Conformational switches as assembly instructions

Let us consider the following scenario of simple one-dimensional assembly. Suppose we are given a one-dimensional component bin which initially contains a random assortment of components. Further suppose we are given a set of assembly instructions, or simply rules, describing which components can bind to which other components. Let the rules be of the form  $a + b \rightarrow ab$ , which means a component  $a$  and a component  $b$  can bind together to form an assembly  $ab$ . Assembly occurs by randomly picking two assemblies<sup>1</sup> in the bin and mating them together. If one of the built-in rules *fires*, the two assemblies can bind together, and the resulting assembly is returned to the bin. To keep track of the subassembly sequences, we parenthesize the resulting assembly when it is formed. The rule  $a + b \rightarrow ab$  fires, for example, when a component  $a$  and a component  $b$  are picked and an assembly  $(ab)$  is added to the bin as a result. If no rules fire, the two assemblies are simply returned to the bin. Note that the rules are assumed to be *local* so that  $a + b \rightarrow ab$  also fires when, for example, an assembly  $(ca)$  and an assembly  $(ba)$  are picked, which results in forming an assembly  $((ca)(ba))$ . This random picking continues until no further rule firing is possible by picking *any* two assemblies in the bin. To see how the above scenario abstracts the one-dimensional self-assembly via sequential random bin-picking discussed in the previous chapter, let us consider a trivial example.

**Example 1** *Suppose our initial bin contains one component  $a$ , one component  $b$ , and two component  $c$ 's (which we represent as  $\langle a, b, c, c \rangle$ ), and our rule set contains  $a + b \rightarrow ab$  and  $b + c \rightarrow bc$ , as shown in Figure 4-1 (a). As we proceed the random picking process described above, no change occurs to the contents of the bin until  $a$  and  $b$  are picked, or  $b$  and  $c$  are picked. If  $a$  and  $b$  are picked, the rule  $a + b \rightarrow ab$  fires and the resulting bin becomes  $\langle (ab), c, c \rangle$  (Figure 4-1 (b1)). After that,  $((ab)c)$  is eventually formed when  $(ab)$  and  $c$  are picked and  $b + c \rightarrow bc$  fires. The resulting bin becomes  $\langle ((ab)c), c \rangle$  and no further rule firing is possible (Figure 4-1 (c1)). Similarly, if  $b$  and  $c$  are picked, the rule  $b + c \rightarrow bc$  fires and the resulting bin becomes  $\langle a, (bc), c \rangle$  (Figure 4-1 (b2)). After that,  $a(bc)$  is formed eventually when  $a$  and  $(bc)$  are picked and  $a + b \rightarrow ab$  fires. The resulting bin becomes  $\langle (a(bc)), c \rangle$ , and no further rule firing is possible (Figure 4-1 (c2)).*

---

<sup>1</sup>we assume a component is a special case of an assembly.

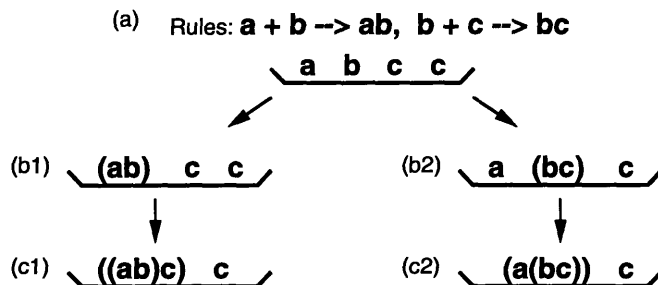


Figure 4-1: One-dimensional self-assembly with no conformational switches:  $abc$  can assemble in the either order  $((ab)c$ ) or  $(a(bc))$ .

In the above example, the rules do not enforce any subassembly sequences to assemble  $abc$ , in other words the final bin contains either  $((ab)c$ ) or  $(a(bc))$  depending on the order of rule firing. One can design conformational switches that enforce  $abc$  to be assembled only in one of the above two subassembly sequences. Since conformational switches are essentially rules of state transition of components triggered by local interaction with other components, we can also represent them as rules of the form  $a + b \rightarrow a'b'$ , which means a component  $a$  and a component  $b$  can bind together to form an assembly  $a'b'$ , where  $a'$  and  $b'$  represents different conformations of  $a$  and  $b$  after conformational changes, respectively. Again, the rules are assumed to be local, hence for example,  $(ca)$  and  $(ba)$  can form  $((ca')(b'a))$  by applying this rule.

**Example 2** Suppose a component  $b$  can take two conformations  $b$  and  $b'$ , and conformational switching between  $b$  and  $b'$  occurs according to the rules  $a + b \rightarrow ab'$  and  $b' + c \rightarrow b'c$ , as shown in Figure 4-2 (a). Note that such rules can be realized using minus devices as described in Section 3.5.3. Starting with the same initial bin,  $\langle a, b, c, c \rangle$ , the random picking process eventually picks up  $a$  and  $b$ . As a result of firing the rule  $a + b \rightarrow ab'$ , the state of the bin becomes  $\langle (ab'), c, c \rangle$  (Figure 4-2 (b)). After that, the rule  $b' + c \rightarrow b'c$  eventually fires to form an  $((ab')c)$ . The resulting bin becomes  $\langle ((ab')c), c \rangle$ , and no further rule firing is possible (Figure 4-2 (c)). Note that conformational change of component  $b$  after binding to  $a$  enforces  $abc$  to be assembled only in the order  $((ab)c)^2$ , by sending out a “signal” that indicates it has bound to  $a$  so it is ready to bind to  $c$ .

<sup>2</sup>Here, we consider  $((ab)c)$  and  $((ab')c)$  are the same assembly.

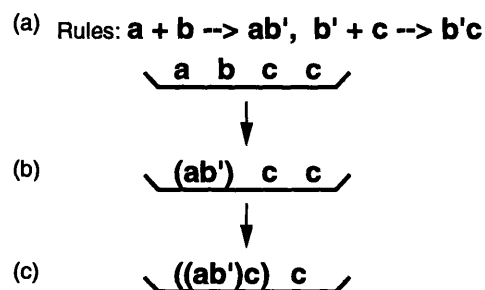


Figure 4-2: One-dimensional self-assembly with conformational switches: components  $abc$  can assemble only in the order  $((ab)c)$ .

Similarly, the rules  $a + b' \rightarrow ab'$ ,  $b + c \rightarrow b'c$  enforces the only possible assembly order to be  $(a(bc))$ .

In Example 2, we can view the rules  $a + b \rightarrow ab'$  and  $b' + c \rightarrow b'c$  as a representation of the subassembly sequence  $((ab)c)$ . In other words, the subassembly sequence  $((ab)c)$  is *encoded* by the conformational switches represented by the rules  $a + b \rightarrow ab'$  and  $b' + c \rightarrow b'c$ . In the following sections, we will discuss which type of conformational switches (equivalently types of the rules which represent the switches) can encode which types of subassembly sequences.

### 4.3 Definition of one-dimensional self-assembling automata

We define a formal model of a one-dimensional self-assembling system as a sequential rule-based machine that operates on one-dimensional strings of symbols. We refer to this machine as a one-dimensional *self-assembling automaton*. In the following, we shall consider a *component* of a one-dimensional self-assembling automaton as an element of a finite set  $\Sigma$ , and an *assembly* is a string in  $\Sigma^+$ . Additionally, a component  $a \in \Sigma$  can take a finite number of *conformations* represented by  $a, a', a'', a''' \dots$ , and the transition from a conformation to the other is triggered by local interactions with other components specified by a set of *assembly rules*<sup>3</sup>.

<sup>3</sup>Each component, therefore, can be viewed as a finite automaton, hence the name self-assembling automata.

**Definition 1** A one-dimensional self-assembling automaton (*henceforth abbreviated SA*) is a pair  $M = (\Sigma, R)$ , where  $\Sigma$  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  $\alpha, \beta, \gamma, \delta \in \{'\}^*$ <sup>4</sup>. The conformation set of  $a \in \Sigma$  is a set  $Q_a = \{a^\alpha \mid \alpha \in \{'\}^*, a^\alpha \text{ appears in } R.\}$ . The conformation set of  $M$  is a union of all conformation sets of  $a \in \Sigma$ .

We will often call a string in  $Q^+$  as an *assembly by conformation*, or simply an *assembly* if there is no ambiguity with a string in  $\Sigma^+$ .

It should be noted that an attaching rule is an abstraction of the function of a minus device, and a propagation rule is an abstraction of one of the functions of a sliding bar mechanism. The other function of a sliding bar mechanism is to detach an existing bond. This can also be abstracted by *detaching rules* of the form  $a^\alpha b^\beta \rightarrow a^\gamma + b^\delta$ . Such detaching rules are not allowed in Definition 1 for simplicity of the discussion. Incorporating detaching rules is certainly a part of future work.

**Example 3** Using the above definition, the self-assembling system in Example 2 can be defined as  $M_1 = (\Sigma, R)$ , where  $\Sigma = \{a, b, c\}$ , and  $R = \{a + b \rightarrow ab', b' + c \rightarrow b'c\}$ . The conformation set of  $M_1$  is  $Q = \{a, b, b', c\}$ .

We view an SA as having an associated *component bin*, with a infinite number of slots each of which can store an assembly (in conformation) or the null string  $\Lambda$ . Initially, a finite number of the slots contain assemblies and the rest of the slots are filled with  $\Lambda$ . Self-assembly of components proceeds by *applying* the rules in  $R$  to the a random pair of assemblies (possibly  $\Lambda$ ) in the component bin. 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 the working memory in rule-based inference systems. The rule application to a random pair  $(x, y)$  is done according to the following procedure:

1. If  $(x, y) = (za^\alpha, b^\beta u)$  for some  $z, u \in \Sigma^*$ , and  $R$  contains the rule  $r = a^\alpha + b^\beta \rightarrow a^\gamma b^\delta$  (which we will refer to as *r fires*), *delete*  $x$  and  $y$  and *add*  $za^\gamma$  and  $b^\delta u$ .

---

<sup>4</sup>It is assumed  $a^\Lambda = a$

2. If  $(x, y) = (za^{\alpha}b^{\beta}u, \Lambda)$  or  $(x, y) = (\Lambda, za^{\alpha}b^{\beta}u)$  for some  $z, u \in \Sigma^+$ , and  $R$  contains the rule  $r = a^{\alpha}b^{\beta} \rightarrow a^{\gamma}b^{\delta}$  (which we will refer to as  $r$  fires), delete  $x$  and  $y$  and add  $za^{\gamma}b^{\delta}u$ .

where  $a, b \in \Sigma$  and  $\alpha, \beta, \gamma, \delta \in \{'\}^*$ . If none of the above applies to  $(x, y)$ ,  $x$  and  $y$  are simply returned to the component bin, leaving the bin unchanged. Note that at any point of self-assembly, the component bin contains a finite number of non-null strings with finite length, since the total number of components in the initial bin is finite and no new component is created by applying the rules to the bin.

In order to describe state of self-assembly at any point, there should be a representation of the component bin which lists all assemblies currently in the component bin. It is also convenient for the representation to keep a record of the sequence in which each assembly has been assembled from its components. To define such a representation, a few notations must be introduced first.

Let  $A$  be a finite set. A *unordered list*  $U$  over a finite set  $A$  is a list of some number of elements in  $A$ , written as  $U = \langle a \mid a \in A \rangle$ . In particular,  $U$  can contain more than one copy of elements in  $A$ . We write  $a \in U$  if  $\text{NUM}_a(U) > 0$ , where  $\text{NUM}_a(U)$  is the number of  $a$ 's in  $U$ . Also, we define  $\text{SEQ}(A)$  to be a shorthand of a language generated by the context-free grammar  $\forall a \in A, S \rightarrow (SS) \mid a$ . Note that  $A \subset \text{SEQ}(A)$ . A string  $x$  in  $\text{SEQ}(A)$  is a full parenthesization of a string  $u = \text{RM-PAREN}(x)$  in  $A^+$ , where  $\text{RM-PAREN}$  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  $\Sigma$  be a component set of an SA. A subassembly sequence is a string in  $\text{SEQ}(\Sigma)$ . A subassembly sequence  $x$  is basic if  $x$  contains at most one copy of elements in  $\Sigma$ , *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 unordered list  $\langle x \mid x \in \text{SEQ}(Q) \rangle$ , where  $Q$  is the conformation set of  $M$ . Let  $x \in \text{SEQ}(\Sigma)$  be a subassembly sequence. A configuration  $\Gamma$  covers  $x$  if  $\Gamma = \langle a \mid a \in \Sigma \rangle$  and  $\forall a \in \Sigma, N_a(x) \leq \text{NUM}_a(\Gamma)$ .

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 non-null string newly added to the component bin are parenthesized in the new configuration if it is added by an attaching rule.

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

**Example 4** *Let us consider  $M_1$  in Example 3. Let  $\Gamma = \langle a, b, c, c \rangle$  and  $\Phi = \langle a, b, c \rangle$ . The configurations  $\Gamma$  and  $\Phi$  covers the subassembly sequence  $((ab)c)$ . Self-assembly of  $((ab)c)$  from  $\Gamma$  proceeds as follows:*

$$\langle a, b, c, c \rangle \vdash_{M_1} \langle (ab'), c, c \rangle \vdash_{M_1} \langle ((ab')c), c \rangle$$

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 contains the string which is assembled in the sequence. Formerly, this can be stated as follows:

**Definition 4** *Let  $M = (\Sigma, R)$  be an SA,  $\Gamma$  be a configuration of  $M$  and  $x \in \text{SEQ}(\Sigma)$  be a subassembly sequence.  $\Gamma$  is stable if there is no rule firing from  $\Gamma$ , i.e.  $C_M(\Gamma) = \{\Gamma\}$ , where  $C_M(\Gamma) = \{\Phi \mid \Gamma \vdash_M^* \Phi\}$ .  $M$  terminates from  $\Gamma$  if all configurations derived from  $\Gamma$  can derive a stable configuration, i.e.  $\forall \Phi \in C_M(\Gamma), \exists \Phi_1 \in C_M(\Phi), C_M(\Phi_1) = \{\Phi_1\}$ .  $M$  self-assembles  $x$  from  $\Gamma$  if both of the following hold:*

1.  $M$  terminates from  $\Gamma$
2.  $\forall \Phi \in C_M^*(\Gamma), \exists y \in \Phi$  such that  $x = \text{RM-PRIME}(y)$ , where  $C_M^*(\Gamma)$  is a set of stable configurations derived from  $\Gamma$ , and  $\text{RM-PRIME}$  is a function that removes the prime symbols ( $'$ ) from its argument.

**Example 5**  $M_1$  in Example 3 self-assembles  $((ab)c)$  from  $\langle a, b, c, c \rangle$ .



## 4.4 Constructing one-dimensional self-assembling automata

Given a basic subassembly sequence  $x \in \text{SEQ}(\Sigma)$ , one can write a procedure which constructs a set of assembly rules  $R$  such that  $M = (\Sigma, R)$  self-assembles  $x$  from any configuration  $\Gamma$  that covers  $x$ . Since  $x$  is a representation of a binary assembly tree, such an algorithm can be written as a simple recursive procedure. The following procedure **MAKE-RULE-SET** takes as input a basic subassembly sequence  $x \in \text{SEQ}(\Sigma)$ , a flag  $\eta \in \{\text{left}, \text{right}, \text{none}\}$ , and a rule set  $R$ . The flag  $\eta$  indicates from which side the next assembly would occur, with *none* indicating there is no next assembly, *i.e.* the current assembly is the last one. **MAKE-RULE-SET**( $x, \text{none}, \emptyset$ ) returns a pair  $(u, R)$ , where  $u$  is the final assembly (by conformation) such that  $\text{RM-PRIME}(u) = \text{RM-PAREN}(x)$  and  $R$  is the rule set containing the assembly rules to assemble  $u$  from  $\Gamma$ . In the following pseudocode<sup>5</sup>,  $x, y, z \in \text{SEQ}(\Sigma)$  are basic subassembly sequences,  $a, b \in \Sigma$ ,  $\alpha, \beta \in \{ '\}^*$ , and  $u, v \in Q^*$  where  $Q = \{a^\alpha \mid a \in \Sigma, \alpha \in \{ '\}^*\}$ , and **LEFT** and **RIGHT** are functions that return the symbol at the left end right end of the argument string, respectively.

```

MAKE-RULE-SET( $x, \eta, R$ )
1  if  $x = a$ 
2    then return  $(a, R)$ 
3  if  $x = (yz)$ 
4    then  $(u, R) \leftarrow \text{MAKE-RULE-SET}(y, \text{right}, R)$ 
5           $(v, R) \leftarrow \text{MAKE-RULE-SET}(z, \text{left}, R)$ 
6           $a^\alpha \leftarrow \text{RIGHT}(u)$ 
7           $b^\beta \leftarrow \text{LEFT}(v)$ 
8          if  $\eta = \text{none}$ 
9            then  $R \leftarrow R \cup \{a^\alpha + b^\beta \rightarrow a^\alpha b^\beta\}$ 
10           return  $(uv, R)$ 
11         if  $\eta = \text{left}$ 
12           then  $R \leftarrow R \cup \{a^\alpha + b^\beta \rightarrow a^{\text{INC}(\alpha)} b^\beta\}$ 
13            $(u, R) \leftarrow \text{PROPAGATE-LEFT}(u, R)$ 
14           return  $(uv, R)$ 

```

---

<sup>5</sup>The pseudocode conventions used in this paper follows [12].

```

15         if  $\eta = right$ 
16             then  $R \leftarrow R \cup \{a^\alpha + b^\beta \rightarrow a^\alpha b^{INC(\beta)}\}$ 
17                  $(v, R) \leftarrow PROPAGATE-RIGHT(v, R)$ 
18             return  $(uv, R)$ 

```

MAKE-RULE-SET recursively traverses the left and right subtrees ( $y$  and  $z$  in the line 3), and adds a attaching rule to  $R$  that assembles ( $yz$ ). If a component will be assembled from the left at the next assembly step ( $\eta = left$  in the line 11), propagation rules are added (by the procedure PROPAGATE-LEFT in the line 13) that propagate conformational changes thorough the assembly corresponds to the left subtree. If a component will be assembled from the right at the next assembly step ( $\eta = right$  in the line 15), propagation rules are added (by the procedure PROPAGATE-RIGHT in line 17) that propagate conformational changes thorough the assembly corresponds to the right subtree. If there is no next assembly step, *i.e.*  $yz$  is the final assembly ( $\eta = none$  in the line 8), no propagation rules need to be added. The subroutines PROPAGATE-LEFT and PROPAGATE-RIGHT are defined as follows:

PROPAGATE-LEFT( $u, R$ )

```

19  if  $u = a^\alpha$ 
20      then return  $(a^{INC(\alpha)}, R)$ 
21  if  $u = va^{\alpha}b^{\beta}$ 
22      then  $R \leftarrow R \cup \{a^{\alpha}b^{INC(\beta)} \rightarrow a^{INC(\alpha)}b^{INC(\beta)}\}$ 
23           $(u, R) \leftarrow PROPAGATE-LEFT(va^{\alpha}, R)$ 
24      return  $(ub^{INC(\beta)}, R)$ 

```

PROPAGATE-RIGHT( $u, R$ )

```

25  if  $u = a^\alpha$ 
26      then return  $(a^{INC(\alpha)}, R)$ 
27  if  $u = a^{\alpha}b^{\beta}v$ 
28      then  $R \leftarrow R \cup \{a^{INC(\alpha)}b^{\beta} \rightarrow a^{INC(\alpha)}b^{INC(\beta)}\}$ 
29           $(u, R) \leftarrow PROPAGATE-RIGHT(b^{\beta}v, R)$ 
30      return  $(a^{INC(\alpha)}u, R)$ 

```

where INC is the “conformation incrementor” function which appends the prime symbol (') to its argument string such that for  $\alpha \in \{\}'^*$ ,  $\text{INC}(\alpha) = \alpha'$ . For example,  $\text{INC}(\Lambda) = \Lambda'$  and  $\text{INC}(\Lambda') = \Lambda''$ .

**Example 6** Let us consider  $\Sigma = \{a, b, c, d\}$  and  $x = ((a(bc))d)$ .  $\text{MAKE-RULE-SET}(x, \text{none}, \emptyset)$  returns with  $(ab''c'd, R)$  where  $R$  contains the following rules:  $b + c \rightarrow b'c$ ,  $a + b' \rightarrow ab''$ ,  $b''c \rightarrow b''c'$  and  $c' + d \rightarrow c'd$ . It is clear that an SA  $M = (\Sigma, R)$  self-assembles  $x$  from the configurations that cover  $x$ , e.g.  $\langle a, b, c, d \rangle$  and  $\langle a, a, b, b, c, c, d, d \rangle$ .

The following theorem tells the correctness of  $\text{MAKE-RULE-SET}$  in general case of  $x$ . Namely, for *any* basic subassembly sequence  $x \in \text{SEQ}(\Sigma)$ ,  $\text{MAKE-RULE-SET}(x, \text{none}, \emptyset)$  returns a pair  $(u, R)$ , where  $\text{RM-PRIME}(u) = \text{RM-PAREN}(x)$  and  $R$  is a set of assembly rules such that an SA  $(\Sigma, R)$  self-assembles  $x$  from any configuration that covers  $x$ .

**Theorem 1**  $\text{MAKE-RULE-SET}$  is correct.

*Proof:* In this proof, we will abbreviate  $\text{MAKE-RULE-SET}$  as  $\text{MRS}$ . Let  $x \in \text{SEQ}(\Sigma)$  be a basic subassembly sequence. we wish to prove the following statement:  $\text{MRS}(x, \text{none}, \emptyset)$  returns  $(u, R)$  such that  $\text{RM-PRIME}(u) = \text{RM-PAREN}(x)$  and  $M = (\Sigma, R)$  self-assembles  $x$  from any configuration  $\Gamma$  that covers  $x$ . The proof is done by the mathematical induction on  $L(x)$ , where  $L(x) = |\text{RM-PAREN}(x)|$ .

**I.** If  $L(x) = 1$ ,  $x = a$ ,  $a \in \Sigma$ .  $\text{MRS}(x, \text{none}, \emptyset)$  immediately returns  $(a, R)$  (the line 2), where  $R = \emptyset$ . Since no assembly is necessary for  $x$ , the above statement is true.

**II.** Suppose the above statement is true for  $L(x) = k$ . In other words, for any  $x_k \in \text{SEQ}(\Sigma)$  and  $L(x) = k$ ,  $M_k = (\Sigma, R_k)$  self-assembles  $x_k$  from any configuration  $\Gamma_k$  that covers  $x_k$ , where  $R_k$  is the rule set returned by  $\text{MRS}(x_k, \text{none}, \emptyset)$ . we are going to construct (by hand)  $R_{k+1}$  from  $R_k$  and then show  $R_{k+1}$  is in fact the rule set returned by  $\text{MRS}(x_{k+1}, \text{none}, \emptyset)$ .

Let  $x_k = (yz)$ ,  $y, z \in \text{SEQ}(\Sigma)$ ,  $(u, R_y) = \text{MRS}(y)$ , and  $(v, R_z) = \text{MRS}(z)$ . Without loss of generality, we can write  $u = a_1^{\alpha_1} a_2^{\alpha_2} \cdots a_i^{\alpha_i}$  and  $v = a_{i+1}^{\alpha_{i+1}} a_{i+2}^{\alpha_{i+2}} \cdots a_k^{\alpha_k}$ , where  $\forall i \in \{1, \dots, k\}$ ,  $a_i \in \Sigma$ ,  $\alpha_i \in \{\}'^*$ . Since we can choose  $x_k$  arbitrarily, any basic subassembly

sequences  $x_{k+1} \in \text{SEQ}(\Sigma)$  with  $L(x_{k+1}) = k + 1$  can be written as either  $(a_{k+1}(yz))$  or  $((yz)a_{k+1})$ , where  $a_{k+1} \in \Sigma$  and  $\forall i \in \{1, \dots, k\}$ ,  $a_i \neq a_{k+1}$ . We consider these two cases separately:

a. If  $x_{k+1} = (a_{k+1}(yz))$ ,  $R_{k+1}$  must contain (1) the rules that assembles  $y$  and  $z$ , (2) the rules that brings  $y$  and  $z$  together, (3) the rules that propagate conformational changes through  $y$  to the left, and (4) the rules that brings  $a_{k+1}$  and  $(yz)$  together. Accordingly, we can construct  $R_{k+1}$  from  $R_k$  by replacing the attaching rule

$$a_i^{\alpha_i} + a_{i+1}^{\alpha_{i+1}} \rightarrow a_i^{\alpha_i} a_{i+1}^{\alpha_{i+1}}$$

with the attaching rule

$$a_i^{\alpha_i} + a_{i+1}^{\alpha_{i+1}} \rightarrow a_i^{\text{INC}(\alpha_i)} a_{i+1}^{\alpha_{i+1}}$$

(this corresponds to (2)), adding the propagation rules

$$\begin{aligned} a_{i-1}^{\alpha_{i-1}} a_i^{\text{INC}(\alpha_i)} &\rightarrow a_{i-1}^{\text{INC}(\alpha_{i-1})} a_i^{\text{INC}(\alpha_i)} \\ a_{i-2}^{\alpha_{i-2}} a_{i-1}^{\text{INC}(\alpha_{i-1})} &\rightarrow a_{i-2}^{\text{INC}(\alpha_{i-2})} a_{i-1}^{\text{INC}(\alpha_{i-1})} \\ &\vdots \\ a_1^{\alpha_1} a_2^{\text{INC}(\alpha_2)} &\rightarrow a_1^{\text{INC}(\alpha_1)} a_2^{\text{INC}(\alpha_2)} \end{aligned}$$

(this corresponds to (3)), and adding the attaching rule

$$a_{k+1} + a_1^{\text{INC}(\alpha_1)} \rightarrow a_{k+1} a_1^{\text{INC}(\alpha_1)}$$

(this corresponds to (4)). Since the rules in  $R_k$  other than  $a_i^{\alpha_i} + a_{i+1}^{\alpha_{i+1}} \rightarrow a_i^{\alpha_i} a_{i+1}^{\alpha_{i+1}}$  are unchanged in  $R_{k+1}$ ,  $R_{k+1}$  contains all the rules that assembles  $x$  and  $y$  separately (this corresponds to (1)).

Now we wish to show that  $R_{k+1}$  constructed as above is in fact the same as the rule set returned by  $\text{MRS}(x_{k+1}, \text{none}, \emptyset)$ . Since  $x_{k+1} = (a_{k+1}(yz))$ ,  $\text{MRS}(x_{k+1}, \text{none}, \emptyset)$  calls  $\text{MRS}(a_{k+1}, \text{right}, \emptyset)$  (the line 4) which immediately returns  $(a_{k+1}, \emptyset)$ , and then calls  $\text{MRS}((yz), \text{left}, \emptyset)$  (the line 5). Let  $\tilde{R}_k$  be the rule set returned by  $\text{MRS}((yz), \text{left}, \emptyset)$ .  $\tilde{R}_k$  is the same as  $R_k$

except that  $\tilde{R}_k$  contains the rule

$$a_i^{\alpha_i} + a_{i+1}^{\alpha_{i+1}} \rightarrow a_i^{\text{INC}(\alpha_i)} a_{i+1}^{\alpha_{i+1}}$$

instead of

$$a_i^{\alpha_i} + a_{i+1}^{\alpha_{i+1}} \rightarrow a_i^{\alpha_i} a_{i+1}^{\alpha_{i+1}}$$

(the line 12), and additionally contains the propagation rules added by PROPAGATE-LEFT (the line 13) which propagate conformational changes through  $y$  to the left. After  $\text{MRS}((yz), \text{left}, \emptyset)$  returns, the rule

$$a_{k+1} + a_1^{\text{INC}(\alpha_1)} \rightarrow a_{k+1} a_1^{\text{INC}(\alpha_1)}$$

is added to  $\tilde{R}_k$  and  $\text{MRS}(x_{k+1}, \text{none}, \emptyset)$  returns. The returned rule set, therefore, contains exactly the same rules as in  $R_{k+1}$  constructed as above.

**b.** If  $x_{k+1} = ((yz)a_{k+1})$ ,  $R_{k+1}$  must contain (1) the rules that assembles  $y$  and  $z$ , (2) the rules that brings  $y$  and  $z$  together, (3) the rules that propagate conformational changes through  $z$  to the right, and (4) the rules that brings  $(yz)$  and  $a_{k+1}$  together. The similar discussion to the part a tells that  $\text{MRS}(x_{k+1}, \text{none}, \emptyset)$  returns the rule set that contains the rules described above. ■

The running time of MAKE-RULE-SET depends on the shape of the parse tree of the input (basic) subassembly sequence. The worst case behavior of MAKE-RULE-SET occurs when, at every step of its recursion, either PROPAGATE-LEFT or PROPAGATE-RIGHT is called. This is the case when new components are added from the alternate directions at every step of assembly. The best case, on the other hand, is when there is no call of PROPAGATE-LEFT and PROPAGATE-RIGHT, *i.e.* at every step of assembly, new components are added from the same directions. The following theorem provides the running time of MAKE-RULE-SET in the worst, the best and the average cases.

**Theorem 2** *Let  $x \in \text{SEQ}(\Sigma)$  be a basic subassembly sequence, and  $n = \text{L}(x)$ . The worst, the best and the average running time of MAKE-RULE-SET( $x, \text{none}, \emptyset$ ) is  $\Theta(n^2)$ ,  $\Theta(n)$  and  $\Theta(n \log n)$ , respectively.*

*Proof:* Let  $x$  be a input string of MAKE-RULE-SET and  $x = (yz)$ . In the worst case scenario, new components are added from the alternate directions at every step of assembly. This implies the input string  $x$  has totally unbalanced subtrees at every recursive step, *i.e.*  $L(y) = 1$  or  $L(y) = n - 1$ . Since PROPAGATE-LEFT and PROPAGATE-RIGHT run in  $\Theta(n)$ , the recurrence of the running time of MAKE-RULE-SET is given as

$$T(n) = T(n - 1) + T(1) + \Theta(n)$$

Since  $T(1) = \Theta(1)$ ,  $T(n) = \Theta(n^2)$ . In the best case, new components are added from the same directions at every step of assembly. This also implies  $L(y) = 1$  or  $L(y) = n - 1$  at every recursive step. Since there is no call of PROPAGATE-LEFT and PROPAGATE-RIGHT, the running time is

$$T(n) = T(n - 1) + T(1) + \Theta(1) = T(n - 1) + \Theta(1) = \Theta(n)$$

On average, we can expect  $L(y) = n/2$  and hence the running time is

$$T(n) = 2T(n/2) + \Theta(n)$$

In the average case, therefore,  $T(n) = \Theta(n \log n)$ . ■

## 4.5 Classes of one-dimensional self-assembling automata

The best running time of MAKE-RULE-SET occurs when neither PROPAGATE-LEFT nor PROPAGATE-RIGHT are called during its execution. In this case, therefore, the rule set  $R$  returned by MAKE-RULE-SET contains *only* attaching rules, whereas *both* attaching rules *and* propagation rules are in  $R$  in other cases. Accordingly, two classes of SA can be defined based on the presence of propagation rules in the rule set.

**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.*

We are going to define the classes of *basic* subassembly sequences which correspond to each of the above classes of SA. The basic subassembly sequences that correspond to the

best running time (hence corresponds to class I SA) are those in which the direction from which new components are added do not alter during the entire self-assembly process. On the other hand, the directions must alter at least once in the basic subassembly sequences that correspond to the worst and average running time (hence corresponds to class II SA). Classes of such subassembly sequences are described more precisely below.

**Definition 6** *An assembly template is a string  $t \in \text{SEQ}(\{p\})$ . An instance of  $t$  on a finite set  $\Sigma$  is a subassembly sequence  $x \in \text{SEQ}(\Sigma)$  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 7** *Two strings  $t_1 = ((pp)(pp))$  and  $t_2 = ((p(pp))p)$  are assembly templates. Let  $\Sigma = \{a, b, c, d\}$ . The basic subassembly sequences  $x_1 = ((ab)(cd))$  and  $x_2 = ((b(ad))c)$  are instances of  $t_1$  and  $t_2$  on  $\Sigma$ , respectively.*

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

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

The assembly templates in  $L(G_I)$  have the structure

$$(((\dots((pp)p)\dots)p)(p(\dots(p(pp))\dots)))$$

whose parse tree is shown in Figure 4-3. Each of the left and right subtrees is a *liner* assembly tree, which specifies self-assembly proceeding in one direction. The parse trees of the assembly templates in  $\text{SEQ}(\{p\})$  are general binary tree with no special structures.

**Example 8** *The assembly template  $t_1$  in Example 7 can be generated by  $G_I$ , for example, through the following derivation:*

$$S \Rightarrow (LR) \Rightarrow ((Lp)R) \Rightarrow ((pp)R) \Rightarrow ((pp)(pR)) \Rightarrow ((pp)(pp))$$

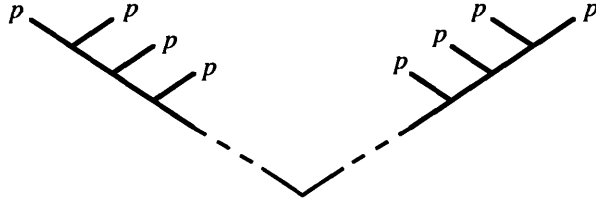


Figure 4-3: Parse tree of an assembly template generated by  $G_I$ .

and hence  $t_1 \in L(G_I)$ .

We can interpret  $L(G_I)$  and  $\text{SEQ}(\{p\})$  as sets of assembly templates with the different number of changes in the direction of self-assembly. Let  $x$  be an subassembly sequence which is an instance of an assembly template  $t$ . If  $t \in L(G_I)$ , the direction of self-assembly does *not* alter during the self-assembly of  $x$ . If  $t \in \text{SEQ}(\{p\}) \setminus L(G_I)$ , the direction of self-assembly alters *at least once* during the self-assembly of  $x$ . Based on these observations, we claim for any basic subassembly sequence whose assembly template is in  $L(G_I)$ , there exists a corresponding class I SA, and for any basic subassembly sequence whose assembly template is in  $\text{SEQ}(\{p\}) \setminus L(G_I)$ , there exists a corresponding class II SA. In the following proofs, we will abbreviate MAKE-RULE-SET as MRS, and  $D(t)$  as the depth of the parse tree of  $t$ .

**Theorem 3** *For any basic subassembly sequence  $x$  which 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  $x$ .*

*Proof:* Let  $x \in \text{SEQ}(\Sigma)$ . By Theorem 1, it suffices to show that the rule set returned by  $\text{MRS}(x, \text{none}, \emptyset)$  contains no propagation rules. If  $D(t) = 0$ ,  $x = a \in \Sigma$ . Therefore,  $\text{MRS}(x, \text{none}, \emptyset)$  immediately returns  $(a, \emptyset)$  (in the line 2). If  $D(t) \geq 1$ , let  $m_l$  and  $m_r$  be the depth of the left and the right subtree of  $t$ , respectively. Without loss of generality we can write  $t = (l_{m_l} r_{m_r})$  where  $l_i = (l_{i-1} p)$  for  $i = 1, \dots, m_l$ ,  $r_i = (p r_{i-1})$  for  $i = 1, \dots, m_r$ , and  $l_0 = r_0 = p$ . Let  $y_i$  and  $z_j$  be substrings of  $x$  that correspond to  $l_i$  and  $r_j$ , respectively. In this case,  $\text{MRS}(x, \text{none}, \emptyset)$  recursively calls  $\text{MRS}(y_{m_l}, \text{right}, \emptyset)$  and  $\text{MRS}(z_{m_r}, \text{left}, R_1)$  (the lines 4 and 5), where  $R_1$  is the rule set returned by  $\text{MRS}(y_{m_l}, \text{right}, \emptyset)$ . Let  $R_2$  be the rule set returned by  $\text{MRS}(z_{m_r}, \text{left}, R_1)$ . Since no propagation rules are added



to  $R_2$  before  $\text{MRS}(x, \text{none}, \emptyset)$  returns in the line 10, it suffices to show neither  $R_1$  nor  $R_2$  contain propagation rules. We will consider these two cases separately.

**A.** To prove  $R_1$  contains no propagation rules, we wish to show that for any  $n \geq 0$ , the rule set  $\tilde{R}_n$  returned by  $\text{MRS}(y_n, \text{right}, \emptyset)$  contains no propagation rules. We will prove the above statement by the mathematical induction on  $n$ .

i. If  $n = 0$ ,  $y_0 = a_0 \in \Sigma$ . Since  $\text{MRS}(a_0, \text{right}, \emptyset)$  returns  $(a_0, \emptyset)$ , no propagation rules are in  $\tilde{R}_0$ .

ii. Suppose for some  $k > 0$ ,  $\text{MRS}(y_k, \text{right}, \emptyset)$  returns with the rule set  $\tilde{R}_k$  which contains no propagation rules. Since  $y_{k+1} = (y_k a_k)$  where  $a_k \in \Sigma$ ,  $\text{MRS}(y_{k+1}, \text{right}, \emptyset)$  recursively calls  $\text{MRS}(y_k, \text{right}, \emptyset)$  and  $\text{MRS}(a_k, \text{left}, \tilde{R}_k)$ . By the inductive hypothesis, no propagation rules are in  $\tilde{R}_k$ . Also,  $\text{MRS}(a_k, \text{left}, \tilde{R}_k)$  returns  $(a_k, \tilde{R}_k)$  since  $a_k \in \Sigma$ . After these calls return, the condition in the line 15 is satisfied and an attaching rule is added to  $\tilde{R}_k$  (the line 16). Let this new rule set be  $\hat{R}$ .  $\text{PROPAGATE-RIGHT}(a_k, \hat{R})$  then is called (the line 17), which returns  $(a'_k, \hat{R})$ . Therefore,  $\text{MRS}(y_{k+1}, \text{right}, \emptyset)$  returns with the rule set  $\tilde{R}_{k+1} = \hat{R}$ , which contains no propagation rules (the line 18).

**B.** To prove  $R_2$  contains no propagation rules, we wish to show that for any  $n \geq 0$ , the rule set  $\tilde{R}_n$  returned by  $\text{MRS}(z_n, \text{left}, R_0)$  contains no propagation rules, where  $R_0$  is a rule set containing no propagation rules. The mathematical induction on  $n$  similar to the part A tells the above statement holds. ■

**Theorem 4** *For any basic subassembly sequence  $x$  which is an instance of an assembly template  $t \in \text{SEQ}(\{p\}) \setminus L(G_I)$ , there exists a class II SA which self-assembles  $x$  from a configuration that covers  $x$ .*

*Proof:* Let  $x \in \text{SEQ}(\Sigma)$ . By Theorem 1, it suffices to show that the rule set returned by  $\text{MRS}(x, \text{none}, \emptyset)$  contains at least one propagation rule. Since

$$\{s \mid s \in \text{SEQ}(\{p\}), D(s) \leq 2\} = \{p, (pp), ((pp)p), (p(pp)), ((pp)(pp))\} \subset L(G_I)$$

we consider  $D(t) \geq 3$ . Let  $m_l$  and  $m_r$  be the depth of the left and the right subtree of  $t$ , respectively. Without loss of generality, we can write  $t = (l_{m_l}^l r_{m_r}^r)$  where  $l_i^l = (l_{i-1}^l l_{i-1}^r)$  for

$i = 1, \dots, m_l$ ,  $r_i^r = (r_{i-1}^l r_{i-1}^r)$  for  $i = 1, \dots, m_r$ ,  $l_i^r, r_i^l \in \text{SEQ}(\{p\})$ , and  $l_0^l = r_0^r = p$ . Then  $\exists j \in \{1, \dots, m_l\}$ ,  $L(l_j^r) \geq 2$ , or  $\exists j \in \{1, \dots, m_r\}$ ,  $L(r_j^l) \geq 2$ , since otherwise  $t \in L(G_I)$ . We consider these two cases separately.

**A.** Suppose  $\exists j \in \{1, \dots, m_l\}$ ,  $L(l_j^r) \geq 2$ . Let  $y_j^l$ ,  $y_j^r$  and  $y_{j+1}^l$ , be substrings of  $x$  that correspond to  $l_j^l$ ,  $l_j^r$  and  $l_{j+1}^l$ . Let  $R_0$  be a rule set containing no propagation rules. It suffices to show the rule set returned by  $\text{MRS}(y_{j+1}^l, \text{right}, R_0)$  contains at least one propagation rule. Since  $y_{j+1}^l = (y_j^l y_j^r)$ ,  $\text{MRS}(y_{j+1}^l, \text{right}, R_0)$  recursively calls  $\text{MRS}(y_j^l, \text{right}, R_0)$  and  $\text{MRS}(y_j^r, \text{left}, R_1)$  (the lines 4 and 5), where  $R_1$  is the rule set returned by  $\text{MRS}(y_j^l, \text{right}, R_0)$ . Let  $(v_j^r, R_2)$  be a return value of  $\text{MRS}(y_j^r, \text{left}, R_1)$ , where  $\text{RM-PRIME}(v_j^r) = \text{RM-PAREN}(y_j^r)$ . Since  $L(y_j^r) \geq 2$ ,  $|v_j^r| \geq 2$ . After  $\text{MRS}(y_j^r, \text{left}, R_1)$  returns, the condition in the line 13 is satisfied and an attaching rule is added to  $R_2$  (the line 16). Let this new rule set be  $\hat{R}_2$ .  $\text{PROPAGATE-RIGHT}(v_j^r, \hat{R}_2)$  then is called (the line 17). Since  $|v_j^r| \geq 2$ , the condition in the line 27 is satisfied and at this point, a propagation rule is added to  $\hat{R}_2$ . Therefore, the rule set returned by  $\text{MRS}(y_{j+1}^l, \text{right}, R_0)$  contains at least one propagation rule.

**B.** Suppose  $\exists j \in \{1, \dots, m_r\}$ ,  $L(r_j^l) \geq 2$ . The similar discussion to the part A tells that the rule set returned by  $\text{MRS}(z_{j+1}^r, \text{left}, R_0)$  contains at least one propagation rule. ■

In addition to the above theorems, we can say that class I SA is not “powerful” enough to encode any basic subassembly sequence which is an instance of an assembly template in  $\text{SEQ}(\{p\}) \setminus L(G_I)$ .

**Corollary 1** *For any basic subassembly sequence  $x$  which is an instance of an assembly template  $t \in \text{SEQ}(\{p\}) \setminus L(G_I)$ , there exist no class I SA which self-assembles  $x$  from a configuration that covers  $x$ .*

*Proof:* By Theorem 4, there exists a class II SA  $M_{II} = (\Sigma, R_{II})$  which self-assembles  $x$  from a configuration that covers  $x$ . Since the conformational changes realized by a propagation rule cannot be realized by using only attaching rules, there are no rule sets containing only attaching rules which is equivalent to  $R_{II}$ . ■

## 4.6 Minimum conformation self-assembling automata

In this section, we will provide the minimum number of conformations necessary to encode a given subassembly sequence based on the classes of basic subassembly sequences introduced in the previous section. Since the number of conformations may vary for each component, we simply define the number of conformations of an SA to be the maximum number of conformations of all components.

**Definition 8** *Let  $M$  be an SA.  $M$  is an SA with  $n$  conformations if*

$$n = \max_{\alpha \in Q} |\alpha|$$

where  $Q$  is the conformation set of  $M$ .

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

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

Note that  $L(G_I) \subset L(G_{II}) \subset \text{SEQ}(\{p\})$ . The assembly templates in  $L(G_{II})$  have the structure

$$((((\dots((R_1 R_1) R_1) \dots) R_1) (L_1 (\dots (L_1 (L_1 L_1)) \dots))))$$

where  $L_1$  and  $R_1$  are strings of the forms  $((\dots((pp)p)\dots)p)$  and  $(p(\dots(p(pp))\dots))$ , respectively. The corresponding parse tree is shown in Figure 4-4. The parse tree in Figure 4-4 can be obtained from the parse tree in Figure 4-3, by replacing leaves at the right branches of the left subtree by a linear assembly tree, and *vice versa*. Let  $x$  be an subassembly sequence and  $t$  is an assembly template of  $x$ . If  $t \in L(G_{II}) \setminus L(G_I)$ , the direction of self-assembly alters *exactly once*, and if  $t \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ , the direction of self-assembly alters *more than once* during the self-assembly of  $x$ .

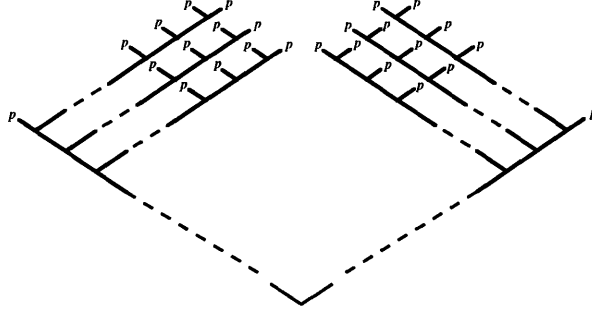


Figure 4-4: Parse tree of an assembly template generated by  $G_{II}$ .

**Example 9** *The assembly template  $t_2$  in Example 7 cannot be generated by  $G_I$  but can be generated by  $G_{II}$ , for example, through the following derivation:*

$$S \Rightarrow (L_0R_0) \Rightarrow ((L_0R_1)R_0) \Rightarrow ((pR_1)R_0) \Rightarrow ((p(pR_1))R_0) \Rightarrow ((p(pp))R_0) \Rightarrow ((p(pp))p)$$

*and hence  $t_2 \in L(G_{II}) \setminus L(G_I)$ . An assembly template  $t_3 = (p((p(pp))p)) \in \text{SEQ}(\{p\})$  cannot be generated by  $G_{II}$  and hence  $t_3 \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ .*

The minimum number of conformations of SA which is necessary to self-assemble a given basic subassembly sequence  $x$  depends on whether  $x$  is an instance of an assembly template in  $L(G_I)$ ,  $L(G_{II}) \setminus L(G_I)$ , or  $\text{SEQ}(\{p\}) \setminus L(G_{II})$ . Since attaching rules produced by MAKE-RULE-SET requires at most two conformations for each component, the minimum number is two if  $x$  is an instance of an assembly template in  $L(G_I)$ . The proof is very similar to the one for Theorem 3.

**Theorem 5** *For any basic subassembly sequence  $x$  which is an instance of an assembly template  $t \in L(G_I)$ , there exists class I SA  $M$  with two (2) conformations which self-assembles  $x$  from a configuration  $\Gamma$  that covers  $x$ . For  $L(x) \geq 3$ ,  $M$  is an SA with the minimum number of conformations which self-assembles  $x$  from  $\Gamma$ .*

*Proof:* Let  $x \in \text{SEQ}(\Sigma)$  and  $R$  be the rule set returned by  $\text{MRS}(x, \text{none}, \emptyset)$ . For the first part, we wish to show the following statement:  $R$  contains only attaching rules of the form  $a^\alpha + b^\beta \rightarrow a^\gamma b^\delta$  such that  $|\alpha|, |\beta|, |\gamma|, |\delta| < 2$ , where  $a, b \in \Sigma$  and  $\alpha, \beta, \gamma, \delta \in \{'\}^*$ . If  $D(t) = 0$ ,  $x = a \in \Sigma$ . Therefore,  $\text{MRS}(x, \text{none}, \emptyset)$  immediately returns  $(a, \emptyset)$  (in the line 2), hence the statement holds. If  $D(t) \geq 1$ , without loss of generality we can write  $t = (l_{m_l} r_{m_r})$

where  $l_i = (l_{i-1}p)$  for  $i = 1, \dots, m_l$ ,  $r_i = (pr_{i-1})$  for  $i = 1, \dots, m_r$ , and  $l_0 = r_0 = p$ . Let  $y_i$  and  $z_j$  be substrings of  $x$  that correspond to  $l_i$  and  $r_j$ , respectively. In this case,  $\text{MRS}(x, \text{none}, \emptyset)$  recursively calls  $\text{MRS}(y_{m_l}, \text{right}, \emptyset)$  and  $\text{MRS}(z_{m_r}, \text{left}, R_1)$  (the lines 4 and 5), where  $R_1$  is the rule set returned by  $\text{MRS}(y_{m-1}, \text{right}, \emptyset)$ . Let  $R_2$  be the rule set returned by  $\text{MRS}(z_{m_r}, \text{left}, R_1)$ . Since there are no INC in the attaching rule added to  $R_2$  in the line 9, it suffices to show the above statement holds for both  $R_1$  and  $R_2$ . We will consider these two cases separately.

**A.** To prove the above statement holds for  $R_1$ , we wish to show that for any  $n \geq 0$ , the statement holds for the rule set  $\tilde{R}_n$  returned by  $\text{MRS}(y_n, \text{right}, \emptyset)$ . We will prove this by the mathematical induction on  $n$ .

i. If  $n = 0$ ,  $y_0 = a_0 \in \Sigma$ . Since  $\text{MRS}(a_0, \text{right}, \emptyset)$  returns  $(a_0, \emptyset)$ , the statement holds for  $\tilde{R}_0 = \emptyset$

ii. Suppose for some  $k > 0$ ,  $\text{MRS}(y_k, \text{right}, \emptyset)$  returns with the rule set  $\tilde{R}_k$  for which the above statement is true. Since  $y_{k+1} = (y_k a_k)$  where  $a_k \in \Sigma$ ,  $\text{MRS}(y_{k+1}, \text{right}, \emptyset)$  recursively calls  $\text{MRS}(y_k, \text{right}, \emptyset)$  and  $\text{MRS}(a_k, \text{left}, \tilde{R}_k)$ . By the inductive hypothesis, the above statement is true for  $\tilde{R}_k$ . Also,  $\text{MRS}(a_k, \text{left}, \tilde{R}_k)$  returns  $(a_k, \tilde{R}_k)$  since  $a_k \in \Sigma$ . After these calls return, the condition in the line 15 is satisfied and an attaching rule  $a_{k-1}^\alpha + a_k \rightarrow a_{k-1}^\alpha a'_k$  is added to  $\tilde{R}_k$ . Let this new rule set be  $\hat{R}$ . Since the above statement holds for  $\tilde{R}_k$ ,  $|\alpha| < 2$ . Therefore, the statement also holds for  $\hat{R}$ .  $\text{PROPAGATE-RIGHT}(a_k, R_1)$  then is called (the line 17), which returns  $(a'_k, R_1)$ . Therefore,  $\text{MRS}(y_{k+1}, \text{right}, \emptyset)$  returns with the rule set  $\tilde{R}_{k+1} = \hat{R}$  for which the above statement holds.

**B.** To prove the above statement holds for  $R_2$ , we wish to show that for any  $n \geq 0$ , the above statement holds for the rule set  $\tilde{R}_n$  returned by  $\text{MRS}(z_n, \text{left}, R_0)$ , where  $R_0$  is a rule set for which the above statement holds. The mathematical induction on  $n$  similar to the part A tells this is the case.

Since at least two conformations are necessary for any  $x$  with  $L(x) \geq 3$ ,  $M$  is an SA with the minimum number of conformations which self-assembles  $x$  from  $\Gamma$ . ■

The ‘‘conformation incrementor’’ INC used in MAKE-RULE-SET simply appends the prime symbol (') to its argument string each time it is called. The number of conformations of a component, therefore, could be very large depending on how many times INC is called

to the component before MAKE-RULE-SET returns. Alternatively, we can use a “modulo  $n$ ” conformation incremator  $\text{INC}_n$  such that

$$\text{INC}_n(\alpha) = \begin{cases} \alpha' & \text{if } |\alpha| < n \\ \Lambda & \text{if } |\alpha| = n \end{cases}$$

For example,  $\text{INC}_2(\Lambda) = \prime$  and  $\text{INC}_2(\prime) = \Lambda$ . Using this notation, we can write INC as  $\text{INC}_\infty$ . Running MAKE-RULE-SET with  $\text{INC}_n$ , instead of  $\text{INC}_\infty$  produces the assembly rules *at most*  $n$  conformations for a component. Such rules, however, are no longer guaranteed to self-assemble the components in a given subassembly sequence. In particular, there could be more than one conflicting propagation rules which specify different conformational changes of the same two adjacent components. In order to show MAKE-RULE-SET run with  $\text{INC}_n$  instead of  $\text{INC}_\infty$  is correct, therefore, it suffices to show no such conflicts among propagation rules are possible. There are two cases to be considered. First, if the rule set  $R$  contains at most one propagation rule for each two adjacent components, no conflict is possible. Therefore, the above statement is true for the smallest possible  $n$ , *i.e.*  $n = 2$ . This is the case if the subassembly sequence  $x$  is an instance of an assembly template  $t \in L(G_{II}) \setminus L(G_I)$ , when the direction of self-assembly alters *exactly once*. Second, if  $R$  contains more than one propagation rules of the same two adjacent components,  $n$  must be large enough to cause no conflicts among the propagation rules. This corresponds to the case where  $x$  is an instance of  $t \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ , when the direction of self-assembly alters *more than once*. In the following proof,  $G_L$  and  $G_R$  are the assembly grammars defined by the production rules  $S \rightarrow (Sp) \mid p$  and  $S \rightarrow (pS) \mid p$ , respectively.

**Theorem 6** *For any basic subassembly sequence  $x$  which is an instance of an assembly template  $t \in L(G_{II}) \setminus L(G_I)$ , there exists class II SA  $M$  with two (2) conformations which self-assembles  $x$  from a configuration  $\Gamma$  that covers  $x$ . And  $M$  is an SA with the minimum number of conformations which self-assembles  $x$  from  $\Gamma$ .*

*Proof:* Let  $x \in \text{SEQ}(\Sigma)$ , and  $R$  be the rule set returned by  $\text{MAKE-RULE-SET}(x, \text{none}, \emptyset)$ . For the first part, we wish to show for any two adjacent components  $ab$  in  $x$ ,  $R$  contains at most one propagation rules of the form  $a^\alpha b^\beta \rightarrow a^\gamma b^\delta$ , where  $\alpha, \beta, \gamma, \delta \in \{\prime\}^*$ ,  $a, b \in \Sigma$  and  $ab$  is a substring in  $\text{RM-PAREN}(x)$ . Without loss of generality, we can write  $t = (l_{m_l}^l r_{m_r}^r)$  where  $l_i^l = (l_{i-1}^l l_{i-1}^r)$  for  $i = 1, \dots, m_l$ ,  $r_i^r = (r_{i-1}^l r_{i-1}^r)$  for  $i = 1, \dots, m_r$ ,  $l_i^l \in L(G_R)$ ,  $r_i^l \in L(G_L)$ ,

and  $l_0^l = r_0^r = p$ . Then  $\exists j \in \{1, \dots, m_l\}, L(l_j^r) \geq 2$ , or  $\exists j \in \{1, \dots, m_r\}, L(r_j^l) \geq 2$ , since otherwise  $t \in L(G_I)$ . We consider these two cases separately.

**A.** Suppose  $\exists j \in \{1, \dots, m_l\}, L(l_j^r) \geq 2$ . Let  $y_j^l, y_j^r$  and  $y_{j+1}^l$ , be substrings of  $x$  that correspond to  $l_j^l, l_j^r$  and  $l_{j+1}^l$ . Let  $ab$  be an arbitrary substring of  $\text{RM-PAREN}(y_{j+1}^l)$ , and  $R_0$  be a rule set containing no propagation rules. Since  $y_{j+1}^l$  and  $y_{j+1}^r$  does not overlap,  $ab$  is not a substring of  $\text{RM-PAREN}(y_{j+1}^r)$ . It suffices, therefore, to show the following statement: the rule set returned by  $\text{MRS}(y_{j+1}^l, \text{right}, R_0)$  contains at most one propagation rules of the form  $a^\alpha b^\beta \rightarrow a^\gamma b^\delta$ , where  $\alpha, \beta, \gamma, \delta \in \{'\}^*$ .

Since  $y_{j+1}^l = (y_j^l y_j^r)$ ,  $\text{MRS}(y_{j+1}^l, \text{right}, R_0)$  recursively calls  $\text{MRS}(y_j^l, \text{right}, R_0)$  and  $\text{MRS}(y_j^r, \text{left}, R_1)$  (the lines 4 and 5), where  $R_1$  is the rule set returned by  $\text{MRS}(y_j^l, \text{right}, R_0)$ . Let  $(v_j^r, R_2)$  be a return value of  $\text{MRS}(y_j^r, \text{left}, R_1)$ , where  $\text{RM-PRIME}(v_j^r) = \text{RM-PAREN}(y_j^r)$ . Since  $l_j^r \in L(G_R)$  and  $L(G_R) \subset L(G_I)$ , only attaching rules are required to assemble  $y_j^r$ , and hence  $R_2$  contains no propagation rules. Since  $L(y_j^r) \geq 2, |v_j^r| \geq 2$ . After  $\text{MRS}(y_j^r, \text{left}, R_1)$  returns, the condition in the line 13 is satisfied and an attaching rule is added to  $R_2$  (the line 16). Let this new rule set be  $\hat{R}_2$ .  $\text{PROPAGATE-RIGHT}(v_j^r, \hat{R}_2)$  then is called (the line 17). Since  $|v_j^r| \geq 2$ , the condition in the line 27 is satisfied and when  $\text{PROPAGATE-RIGHT}(v_j^r, \hat{R}_2)$  returns, exactly one propagation rule of the form  $a^\alpha b^\beta \rightarrow a^\alpha b(\text{INC}(\beta))$  is added to  $\hat{R}_2$  for each substring  $ab$  of  $\text{RM-RAREN}(y_j^r)$ . Since this is the only time the propagation rules are added, the above statement holds.

**B.** Suppose  $\exists j \in \{1, \dots, m_r\}, L(r_j^l) \geq 2$ . Let  $ab$  be an arbitrary substring of  $\text{RM-PAREN}(z_{j+1}^r)$ , The similar discussion to the part A tells that the rule set returned by  $\text{MRS}(z_{j+1}^r, \text{left}, R_0)$  contains at most one propagation rule of the form  $a^\alpha b^\beta \rightarrow a^\gamma b^\delta$  for each substring  $ab$  of  $\text{RM-RAREN}(z_j^l)$ .

$\text{MAKE-RULE-SET}(x, \text{none}, \emptyset)$  run with  $\text{INC}_2$  causes no conflict among propagation rules, since  $R$  contains at most one propagation rules for any two adjacent components in  $x$ . By Theorems 1 and 4, therefore, there exist a class II SA  $M$  with two conformations which self-assembles  $x$  from  $\Gamma$ . Since  $L(G_I) \subset L(G_{II})$ , Theorem 5 tells at least two conformations are necessary and therefore,  $M$  is an SA with minimum conformation which self-assembles  $x$  from  $\Gamma$ . ■

**Example 10** Let us consider  $\Sigma = \{a, b, c, d\}$  and  $x = ((a(bc))d)$ . The subassembly sequence  $x$  is an instance of  $t_2 = ((p(pp))p)$  in Example 7. From Example 9,  $t_2 \in L(G_{II}) \setminus L(G_I)$ . A call of  $\text{MAKE-RULE-SET}(x, \text{none}, \emptyset)$  run with  $\text{INC}_2$  returns with  $(abc'd, R)$  where  $R$  contains 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  $\langle a, a, b, b, c, c, d, d \rangle$ .

In the case where a basic subassembly sequence  $x$  is an instance of assembly template in  $\text{SEQ}(\{p\}) \setminus L(G_{II})$ , we claim only *three* conformations are necessary to encode arbitrary  $x$ . This might sound counter-intuitive since we are claiming only three conformations can encode basic subassembly sequences with arbitrary (possibly *very* large) sizes. The proof of this claim is based on the observation that there are only two kinds of propagation rules; the rules which propagate conformational changes to the left, and the rules which propagates conformational changes to the right. As in the previous case, we will prove this statement by showing  $\text{MAKE-RULE-SET}$  run with  $\text{INC}_n$  causes no conflicts among propagation rules of the same adjacent components in the case of  $n = 3$ . To do this, we will define a concept called a  $n$ -conformation transition cycle. Then, we will prove that no such conflicts among propagation rules are possible for  $\text{INC}_n$  if there exists a  $n$ -conformation transition cycle. Finally, we will show there exists a 3-conformation transition cycle. Since our focus is on conformational propagation between two arbitrary adjacent components, it is convenient to introduce several notations first.

A *conformational transition rule* is a rule of the form  $\alpha \cdot \beta \rightarrow \gamma \cdot \delta$ , where  $\alpha, \beta, \gamma, \delta \in \{\prime\}^*$ . Let  $r$  be a propagation rule and  $\rho$  be a conformational transition rule. We write  $\rho = \text{TRN}(r)$  if  $\rho = \alpha \cdot \beta \rightarrow \gamma \cdot \delta$  and  $r = a^\alpha b^\beta \rightarrow a^\gamma b^\delta$  where  $a, b \in \Sigma$ . For two conformational transition rules  $\rho_1$  and  $\rho_2$ , we write  $\rho_1 \rightsquigarrow_n \rho_2$  if *one* of the following holds:

- $\rho_1 = \alpha \cdot \beta \rightarrow \text{INC}_n(\alpha) \cdot \beta$  and  $\rho_2 = \text{INC}_n(\text{INC}_n(\alpha)) \cdot \beta \rightarrow \text{INC}_n(\text{INC}_n(\alpha)) \cdot \text{INC}_n(\beta)$ .
- $\rho_1 = \alpha \cdot \beta \rightarrow \alpha \cdot \text{INC}_n(\beta)$  and  $\rho_2 = \alpha \cdot \text{INC}_n(\text{INC}_n(\beta)) \rightarrow \text{INC}_n(\alpha) \cdot \text{INC}_n(\text{INC}_n(\beta))$ .

In addition, we say two conformational transition rules  $\rho_1 = \alpha_1 \cdot \beta_1 \rightarrow \gamma_1 \cdot \delta_1$  and  $\rho_2 = \alpha_2 \cdot \beta_2 \rightarrow \gamma_2 \cdot \delta_2$  are *conflicting* if  $(\alpha_1, \beta_1) = (\alpha_2, \beta_2)$  and  $(\gamma_1, \delta_1) \neq (\gamma_2, \delta_2)$ .



**Definition 10** A  $n$ -conformation transition cycle is a finite sequence of non-conflicting conformational transition rules  $\langle \rho_1, \rho_2, \dots, \rho_m \rangle$  such that for  $i = 1, 2, \dots, m-1$ ,  $\rho_i \rightsquigarrow_n \rho_{i+1}$ , and  $\rho_m \rightsquigarrow_n \rho_1$ .

**Corollary 2** Let  $x \in \text{SEQ}(\Sigma)$  be basic subassembly sequence which is an instance of an assembly template  $t \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ . Let  $R$  be the rule set returned by  $\text{MAKE-RULE-SET}(x, \eta, \emptyset)$  run with  $\text{INC}_\infty$ . There exist a substring  $ab$  of  $\text{RM-PAREN}(x)$  such that  $R$  contains more than one propagation rules of  $ab$ . For any two propagation rules  $r_1, r_2 \in R$  of  $ab$ ,  $\text{TRN}(r_1) \rightsquigarrow_\infty \text{TRN}(r_2)$  if  $r_2$  fires after  $r_1$  and no propagation rules of  $ab$  fires between  $r_1$  and  $r_2$ .

*Proof:* Let  $\alpha, \beta \in \{ '\}^*$ . During the self-assembly of  $x$ , the direction of self-assembly alters more than once since otherwise  $t \in L(G_{II}) \setminus L(G_I)$ . This implies there exist at least one substring  $ab$  of  $\text{RM-PAREN}(x)$  such that  $R$  contains more than one propagation rules of the form  $a^\alpha b^\beta \rightarrow a^\gamma b^\delta$ .

If  $r_1$  is added to  $R$  by  $\text{PROPAGATE-LEFT}$  in the line 22, we can write  $r_1$  as  $r_1 = a^\alpha b^\beta \rightarrow a^{\text{INC}_\infty(\alpha)} b^\beta$ . We are going to show  $r_2$  must be then added to  $R$  by  $\text{PROPAGATE-RIGHT}$ . Let us suppose  $r_2$  is added to  $R$  by  $\text{PROPAGATE-LEFT}$ . This implies  $\text{PROPAGATE-LEFT}$  is called twice in the two consecutive assembly steps. This then implies two components are added from the left in the two consecutive assembly steps, since  $\text{PROPAGATE-LEFT}$  is called when the component at the next assembly step is added from the left. If this is the case, however, the second call of  $\text{PROPAGATE-LEFT}$  returns in the line 20, without adding any propagation rules to  $R$ . This is contradiction. Therefore,  $r_2$  must be added to  $R$  by  $\text{PROPAGATE-RIGHT}$ . Since  $r_1 = a^\alpha b^\beta \rightarrow a^{\text{INC}_\infty(\alpha)} b^\beta$  and no propagation rules of  $ab$  fires between  $r_1$  and  $r_2$ ,  $r_2$  must be the form  $r_2 = a^{\text{INC}_\infty(\text{INC}_\infty(\alpha))} b^\beta \rightarrow a^{\text{INC}_\infty(\text{INC}_\infty(\alpha))} b^{\text{INC}_\infty(\beta)}$ . Hence  $\text{TRN}(r_1) \rightsquigarrow_\infty \text{TRN}(r_2)$ .

If  $r_1$  is added to  $R$  by  $\text{PROPAGATE-RIGHT}$  in the line 29, we can write  $r_1$  as  $r_1 = a^\alpha b^\beta \rightarrow a^\alpha b^{\text{INC}_\infty(\beta)}$ . Similar discussion shows  $r_2$  must be added to  $R$  by  $\text{PROPAGATE-LIGHT}$ , and  $r_2$  must be the form  $r_2 = a^\alpha b^{\text{INC}_\infty(\text{INC}_\infty(\beta))} \rightarrow a^{\text{INC}_\infty(\alpha)} b^{\text{INC}_\infty(\text{INC}_\infty(\beta))}$ . Hence in this case also,  $\text{TRN}(r_1) \rightsquigarrow_\infty \text{TRN}(r_2)$ . ■

**Corollary 3** Let  $n \geq 3$ . For any basic subassembly sequence  $x$  which is an instance of an assembly template  $t \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ , there exist a class II SA with  $n$  conformations which self-assembles  $x$  from a configuration that covers  $x$  if there exists a  $n$ -conformation transition cycle.

*Proof:* Let  $x \in \text{SEQ}(\Sigma)$ . From Theorem 1, it suffices to show that MAKE-RULE-SET run with  $\text{INC}_n$ , is correct. Let  $R_\infty$  and  $R_n$  be the rule sets returned by MAKE-RULE-SET( $x, \text{none}, \emptyset$ ) run with  $\text{INC}_\infty$  and with  $\text{INC}_n$ , respectively. We know  $R_\infty$  and  $R_n$  contain exactly the same attaching rules since  $n \geq 3$  and attaching rules increment conformation of a component at most twice. Let  $\langle r_\infty^1, r_\infty^2, \dots, r_\infty^k \rangle$  be a sequence of propagation rules of two adjacent components  $ab$ , as produced by MAKE-RULE-SET( $x, \text{none}, \emptyset$ ) with  $\text{INC}_\infty$ . Also, let  $\langle r_n^1, r_n^2, \dots, r_n^l \rangle$  be a sequence of propagation rules of  $ab$ , as produced by MAKE-RULE-SET( $x, \text{none}, \emptyset$ ) with  $\text{INC}_n$ , where  $l \leq k$ . By definition of  $\text{INC}_n$ ,  $r_\infty^i$  corresponds to  $r_n^{i \bmod l}$  for  $i = 1, 2, \dots, k$ . From Corollary 2, therefore,  $\text{TRN}(r_n^{i \bmod l}) \rightsquigarrow_n \text{TRN}(r_n^{(i+1) \bmod l})$  for  $i = 1, 2, \dots, k-1$ . Since there exists a  $n$ -conformation transition cycle,  $\text{TRN}(r_n^i)$  and  $\text{TRN}(r_n^j)$  are not conflicting for any  $i, j \in \{1, 2, \dots, l\}$ . Since this holds for propagation rules of any two adjacent components,  $R_n$  contains the rules which assembles  $x$  from a configuration that covers  $x$ . ■

**Corollary 4** *There exists a 3-conformation propagation cycle.*

*Proof:* In this proof, we will write the prime symbol ( $'$ ) as  $p$ . Let us consider a sequence of conformational transition rules  $\langle r_1, r_2, r_3, r_4, r_5, r_6 \rangle$  where

$$\begin{array}{lll} r_1 = \Lambda \cdot \Lambda \rightarrow p \cdot \Lambda & r_2 = pp \cdot \Lambda \rightarrow pp \cdot p & r_3 = pp \cdot pp \rightarrow \Lambda \cdot pp \\ r_4 = p \cdot pp \rightarrow p \cdot \Lambda & r_5 = p \cdot p \rightarrow pp \cdot p & r_6 = \Lambda \cdot p \rightarrow \Lambda \cdot pp \end{array}$$

By Definition 10, the above sequence is a 3-conformation propagation cycle, since  $\text{INC}_3(\Lambda) = p$ ,  $\text{INC}_3(p) = pp$ , and  $\text{INC}_3(pp) = \Lambda$ . ■

**Theorem 7** *For any basic subassembly sequence  $x$  which is an instance of an assembly template  $t \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ , there exists class II SA  $M$  with three (3) conformations which self-assembles  $x$  from a configuration  $\Gamma$  that covers  $x$ . And  $M$  is an SA with the minimum number of conformations which self-assembles  $x$  from  $\Gamma$ .*

*Proof:* The first part follows from Corollaries 3 and 4. We will prove the second part by showing that there exists no class I SA with 2 conformations which self-assembles  $x$  from  $\Gamma$ . Since  $\text{INC}_2(\text{INC}_2(\alpha)) = \alpha$  for  $\alpha \in \{'\}^*$ , any two conformational transition rules  $\rho_1$  and  $\rho_2$

are conflicting if  $\rho_1 \rightsquigarrow_2 \rho_2$ . By Corollary 2, therefore, running MAKE-RULE-SET with INC<sub>2</sub> always causes at least one conflict among propagation rules, since  $t \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ . ■

**Example 11** *Let us consider  $\Sigma = \{a, b, c, d, e\}$  and  $x = (a((b(cd))e))$ . The subassembly sequence  $x$  is an instance of  $t_3 = (p((p(pp))p))$  in example Example 9, and  $t_3 \in \text{SEQ}(\{p\}) \setminus L(G_{II})$ . A call of MAKE-RULE-SET( $x, \text{none}, \emptyset$ ) run with INC<sub>3</sub> returns with  $(ab'c'd''e, R)$  where  $R$  contains 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  $\langle a, b, b, c, d, d, e, e \rangle$ .*

## 4.7 Summary

This chapter introduced an abstract model of self-assembling systems, where assembly instructions of components are written as conformational switches – local rules that specify conformational changes of a component. The model, self-assembling automaton, is a sequential rule-based machine that operates on one-dimensional strings of symbols. An algorithm to construct a self-assembling automaton is provided which self-assembles an one-dimensional string of distinct symbols in a given particular subassembly sequence. Classes of self-assembling automata are then defined based on classes of subassembly sequences in which the components self-assemble. For each class of subassembly sequences, I provided the minimum number of conformations necessary to encode subassembly sequences of the class. In particular, it is shown that attaching rules and propagation rules, which are abstraction of minus devices and sliding bar mechanisms, as well as three conformations for each component, is enough to encode any subassembly sequences of a string with arbitrary length.

---

## Conformational Switch for Micro Assembly: Micro “Mouse Trap”

---

This chapter presents an implementation of conformational switch for micro assembly – a micro “mouse trap,” which has self-closing compliant latches that clamp a free planar part inserted between them. The self-closing of the latches is induced by the insertion of the part which releases the potential energy stored in the “cocked” state of the device, in the manner of a mouse trap. The closed latch can re-open by cocking the micro mouse trap. The self-closing action of the micro mouse trap allows the inaccurate insertion of the free part (mouse) and leads to very accurate final positioning, with the insertion force required to induce the self-closing being very small. The micro mouse trap is also reusable since the release of the clamped part is done non-destructively. The micro mouse trap can be used, for instance, for assembly and temporal clamping/positioning of micro-scale parts, and electromechanical/opto-mechanical connectors. A prototype of the micro mouse trap, which is approximately 150  $\mu m$  long, 200  $\mu m$  wide, and 10  $\mu m$  thick, is fabricated from single crystal silicon. The prototype device is tested using probe tips and the desired self-closing behavior is observed.

## 5.1 Motivation

Surface etching technologies have been widely used in micro fabrication, and a number of micro devices have been successfully fabricated using these technologies. One of the advantages of using surface etching in micro fabrication is there is no need for assembly; the entire device with multiple components can be fabricated in one process by using sacrificial layers. The design of micro devices, therefore, has focused on minimization or elimination of assembly. One can imagine, however, that there is a natural limit in complexity for micro electro-mechanical systems constructed without assembly. Beyond the limit, some assembly methods would be necessary and these methods would have characteristics different from the assembly methods of macro scale components. For instance, precise handling and positioning of components involved in typical macro scale assembly is extremely difficult in micro scale. One solution to this problem is to make components that are capable of positioning themselves as observed in the *induced fit* in enzyme-substrate interactions. An enzyme molecule binds to a substrate molecule by changing its conformation such that it can “grab” the substrate. The conformational change of enzyme molecule is induced by the interaction with the substrate molecule which causes the release of potential energy stored in high-energy molecules such as ATP [58].

The design of a micro “mouse-trap” presented in this chapter is the first step toward realizing mechanical systems with such induced fit capability. The micro mouse trap is a micro latching fastener with self-closing compliant latches that clamp a free planar part inserted between them. The self-closing of the latches is induced by the insertion of the part which releases the potential energy stored in the “cocked” state of the device. The self-closing action of the micro mouse trap allows the inaccurate insertion of the free part (mouse) that leads to very accurate final positioning.

## 5.2 Mouse trap design

The micro mouse trap is a doubly-nested compliant crimping mechanism [4] with a snap fit stopper, as shown in Figure 5-1. The device consists of three parts: outer and inner latches (mouse trap), and a free part (mouse), as shown in Figure 5-1 (1),(2) and (3) respectively. The free part is a part of a substrate-free structure to be clamped by the mouse trap, and

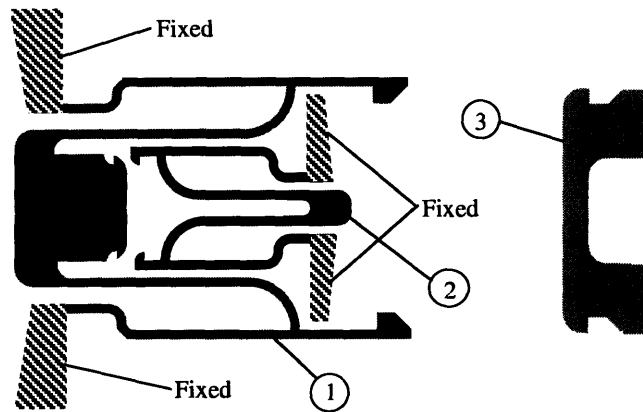


Figure 5-1: Schematic top view of the micro mouse trap. The free part (mouse) (3) is clamped by the outer latch (1).

assumed to be able to slide on the substrate. The outer and inner latches are cantilever structures anchored on the substrate at the locations shown in Figure 5-1.

Figure 5-2 shows the four steps of the action of the micro fastening device: (a) cock, (b) insertion, (c) clamping, and (d) releasing. Before clamping the free part, the outer latch is “cocked” by snapping *A* into the inner latch *B*, as shown in Figure 5-2 (a). The cocked outer latch is shown in Figure 5-2 (b). Once cocked, the outer latch opens up so that the free part can be inserted. At full insertion of the free part, *C*’s pressing *D* causes the release of *A* from *B*, causing the outer latch to clamp the free part (Figure 5-2 (c)). The clamped part can be released by re-cocking the outer latch (Figure 5-2 (d)).

The self-closing of the latches in Figure 5-2 (c) is induced by the insertion of the free part that releases the potential energy stored in the cocked state of the device, in the manner of a mouse trap. The above design of a micro mouse trap has the following properties desired for micro assembly:

- **Mechanical fastening:** the free part is mechanically clamped by the outer latch, which provides a fastening method for materials that would be incompatible with other methods, *e.g.* chemical bonding.
- **Self-positioning:** the self-closing action of the micro mouse trap allows the inaccurate insertion of the free part that leads to very accurate final positioning.

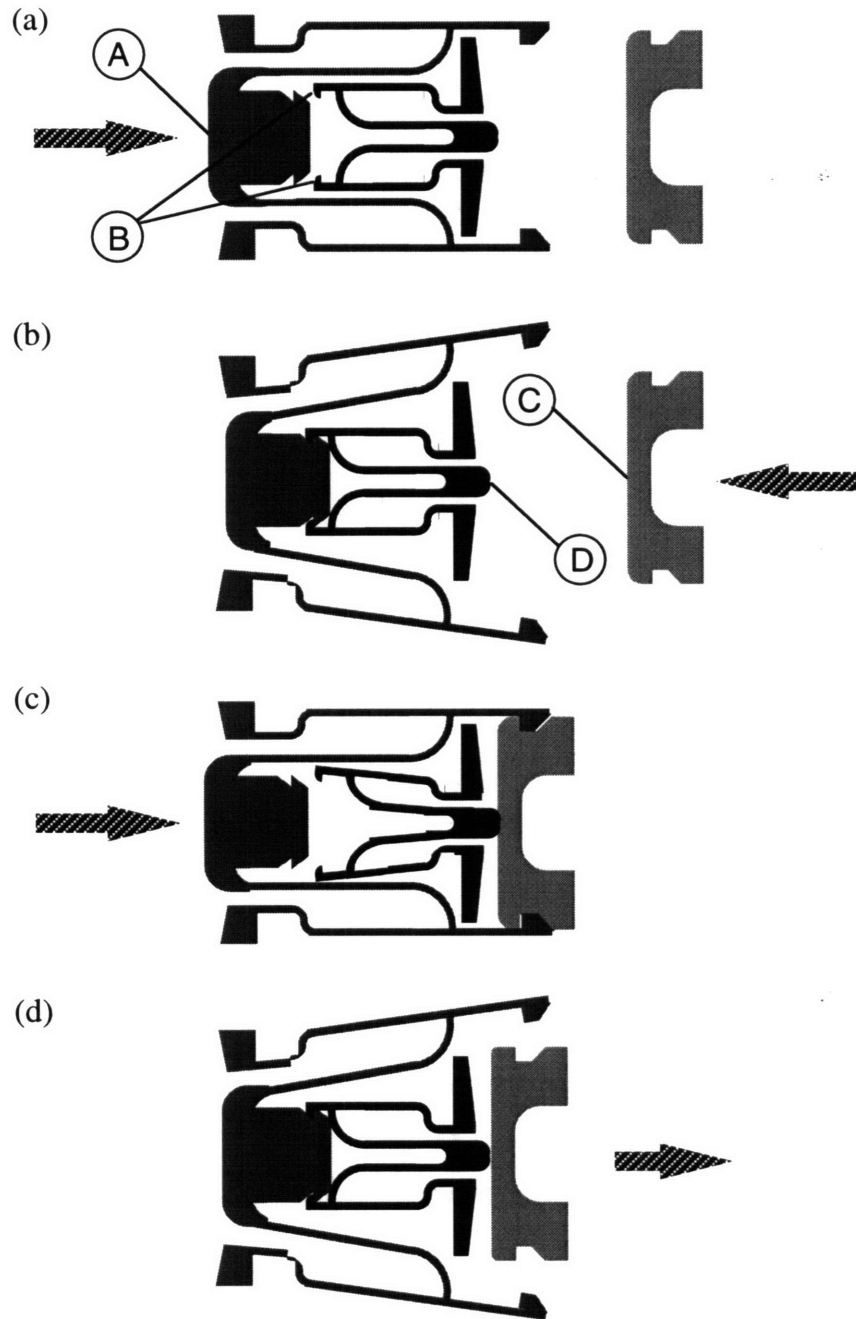


Figure 5-2: The four steps of the action of the micro mouse trap; (a) cock, (b) insertion, (c) clamping and (d) releasing. (a) The outer latch is “cocked” by snapping *A* into the inner latch *B*. (b) The cocked outer latch opens up its arms so that the free part can be inserted. (c) *C*’s pressing *D* causes the release of *A* from *B*, causing the outer latch to clamp the free part. (d) The clamped part is released by re-cocking the outer latch.

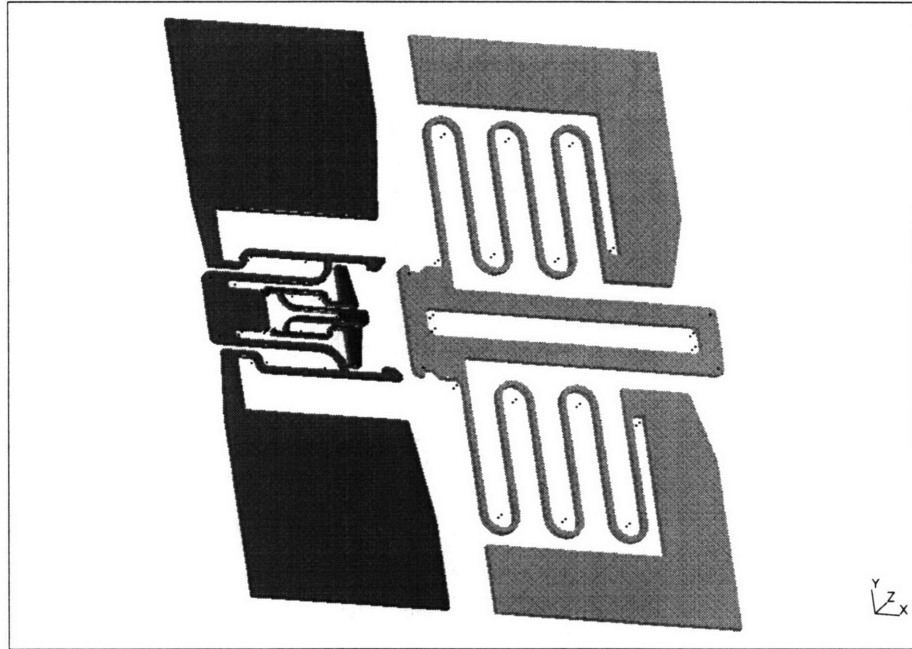


Figure 5-3: Solid model for the prototype micro mouse trap. The size of the prototype (excluding anchors and the free part) is approximately  $250\ \mu\text{m}$  long,  $150\ \mu\text{m}$  wide, and  $10\ \mu\text{m}$  thick. The gap between the substrate and the cantilevered structure is  $5\ \mu\text{m}$ .

- **Force amplification:** the insertion force required to induce the self-closing of the outer latch (by opening the inner latch) is much smaller than the actual clamping force exerted by the outer latch.
- **Reusability:** the micro mouse trap is reusable since the release of the clamped part is done non-destructively.
- **Ease of fabrication:** since the micro fastening device is a cantilevered compliant mechanism, it is easily fabricated by normal surface etching processes, making the integration with electrical components very easy.

A solid model of the prototype micro mouse trap was constructed with the SDRC I-DEAS system to perform finite element structural analysis. Figure 5-3 shows a view of the solid model used for the structural analysis. The prototype is designed to be fabricated on a single crystal silicon (SCS) substrate, and to use microprobe tips to demonstrate the desired function described above. As shown in Figure 5-3, the solid model has larger anchors than shown in Figure 5-1 for easier alignment for silicon-to-silicon wafer bonding described in the next section. Also, for demonstration purposes, the free part is supported by a



cantilevered serpentine to allow insertion from various positions and orientations. The size of the prototype (excluding anchors and the free part) is approximately  $250\ \mu\text{m}$  long,  $150\ \mu\text{m}$  wide, and  $10\ \mu\text{m}$  thick. The gap between the substrate and the cantilevered structure is  $5\ \mu\text{m}$ . The material properties of SCS used in the following FEM analyses were taken from [43].

Figure 5-4 shows the predicted deflection of the outer latch<sup>1</sup> in response to a compressive force (pointing to the right) of  $1000\ \mu\text{N}$  exerted at the left end of the outer latch (*i.e.* “cocking”). The outer latch opens approximately  $60\ \mu\text{m}$ , which gives a enough room for the free part for insertion.

Figure 5-5 shows the predicted deflection of the inner latch in response to an extensile force (pointing to the left) of  $1000\ \mu\text{N}$  exerted at the hook of the inner latch. This is the situation shown in Figure 5-2 (b), after the outer latch is cocked by snapping part *A* into the inner latch *B*. The S-shape of the beam generates a counterclockwise bending moment in response to the extensile force, which causes the inner latch to close rather than open. Similarly, the FEM result of the situation in Figure 5-2 (c) is shown in Figure 5-6, where a compressive force of  $200\ \mu\text{N}$  (pointing to the left) is exerted at the right end of the inner latch, *in addition to* the extensile force of  $1000\ \mu\text{N}$  (pointing to the left) at the hook of the inner latch. No friction is assumed between the two surfaces at the hook . In this case, the inner latch opens approximately  $6\ \mu\text{m}$ , which is enough to release the hook.

In the above FEM results, a maximum stress of approximately  $2.5\ \text{GPa}$  (35% of the yield stress of SCS) occurred near the S-shaped part of the outer latch and the inner latch.

### 5.3 Device fabrication

A combination of Plasma etching, reactive ion etching (wet etching) and a silicon-to-silicon wafer bonding technique is used for fabrication of the prototype of the mouse trap device. Two silicon (Si) wafers, which is referred to as the base wafer and the structure wafer, are used for the substrate of the device and the device itself, respectively. The fabrication process consists of the following eight steps (see also Figure 5-7):

---

<sup>1</sup>Displacements are shown in actual scale.

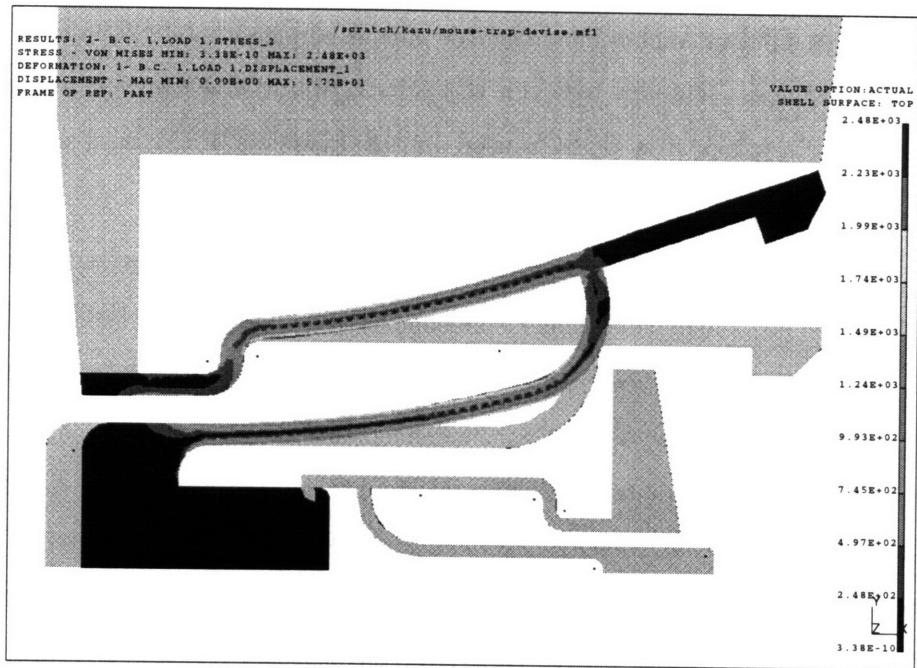


Figure 5-4: FEM result of the outer latch of the prototype mouse trap. A compressive force of  $1000\ \mu\text{N}$  (pointing to the right) is exerted at the left end of the outer latch.

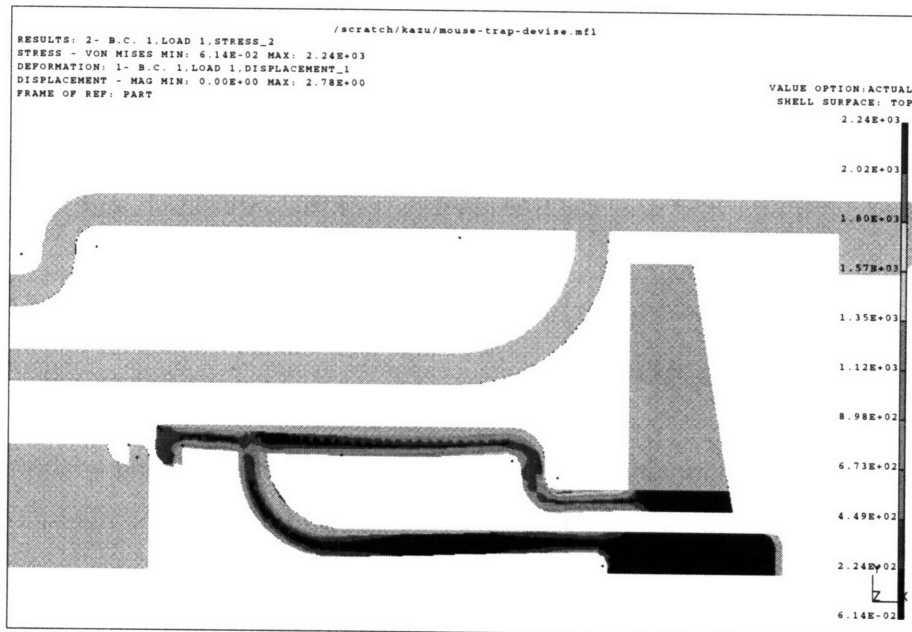


Figure 5-5: FEM result of the inner latch of the prototype mouse trap. An extensile force of  $1000\ \mu\text{N}$  (pointing to the left) is exerted at the hook of the inner latch.

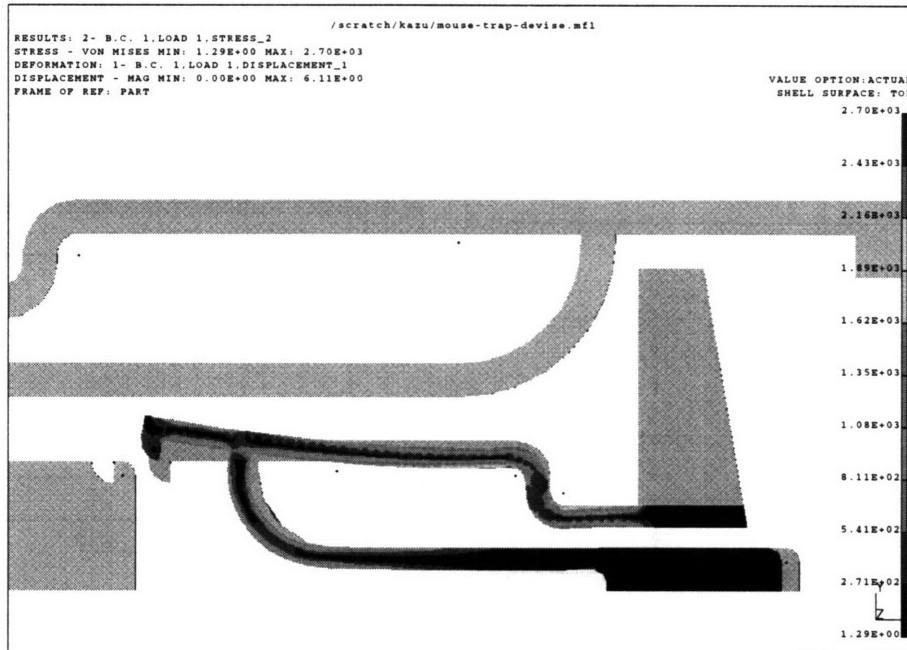


Figure 5-6: FEM result of the inner latch of the prototype mouse trap. A compressive force of  $200 \mu N$  (pointing to the left) is exerted at the right end of the inner latch, as well as an extensile force of  $1000 \mu N$  (pointing to the left) at the hook of the inner latch.

1. Plasma etch the base wafer with a cavity mask to create the  $5 \mu m$  deep cavity and the anchors.
2. Bond the base wafer and an silicon-on-insulator (SOI) wafer (a sandwich of silicon dioxide with two Si wafers) at the position of the anchors using silicon-to-silicon wafer bond at  $1100^\circ C$ .
3. Deposit silicon nitride on the bonded wafers using low-pressure chemical vapor deposition (LPCVD).
4. Plasma etch silicon nitride on the top of the bonded wafers.
5. Wet etch (KOH) the top layer of Si in the SOI wafer to expose the silicon dioxide layer.
6. Wet etch (buffered oxide etch: BOE) the silicon dioxide layer.
7. Wet etch silicon nitride on the bottom.
8. Plasma etch the structure wafer with a structure mask to create the actual shape of the device with  $10 \mu m$  thick.

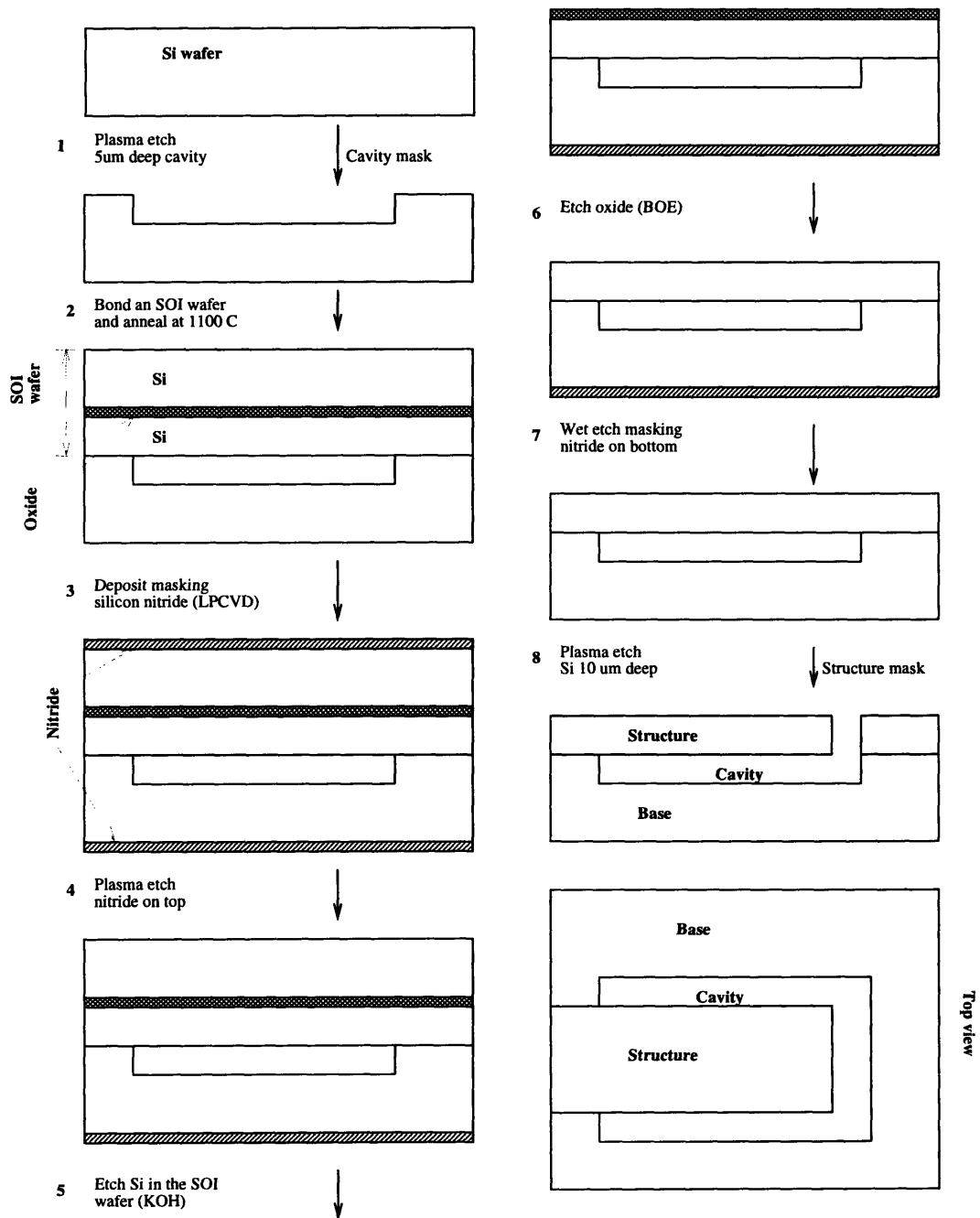


Figure 5-7: Fabrication process flow: courtesy of G.K Ananthasuresh at the MIT Microsystems Technology Laboratory.

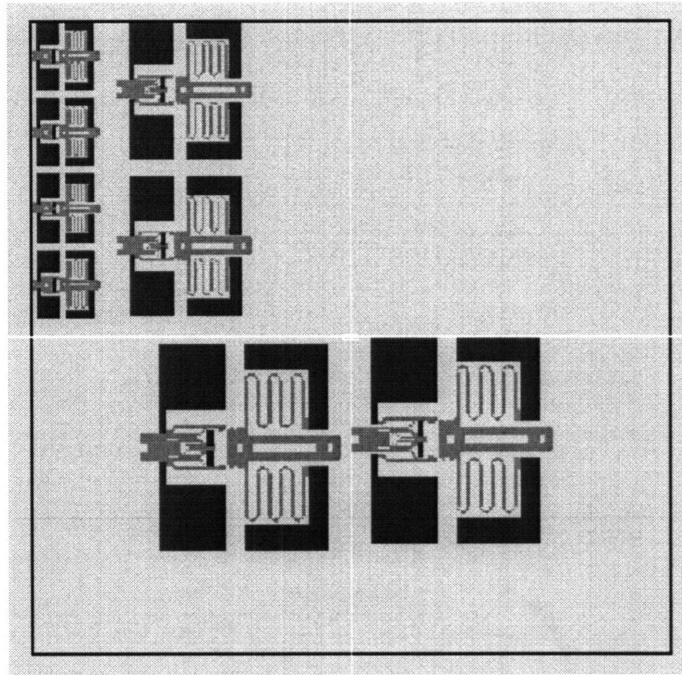


Figure 5-8: Structure mask layout: the devices in the original size ( $150 \times 200 \mu m$ ), the double size, and the quadruple size are shown. Courtesy of G.K Ananthasuresh at the MIT Microsystems Technology Laboratory.

The layout of the structure mask used at Step 8 is shown in Figure 5-8. The devices in the original size ( $150 \times 200 \mu m$ ), the double size, and the quadruple size are shown. Figure 5-9 shows an SEM micrograph of the fabricated micro mouse trap. It is approximately  $150 \mu m$  long,  $200 \mu m$  wide, and  $10 \mu m$  thick. This prototype device is tested using probe tips and the desired self-closing behavior is observed.

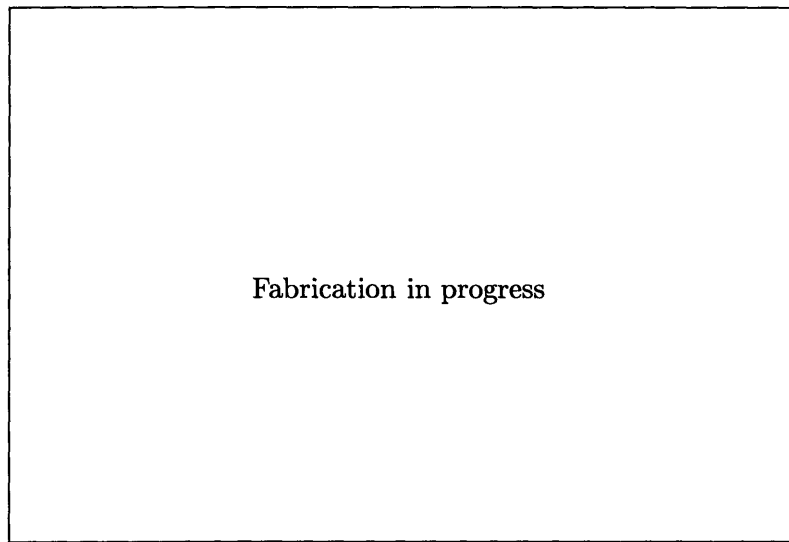


Figure 5-9: Prototype micro mouse trap (SEM micrograph). The device is tested using probe tips and the desired self-closing behavior is observed.

---

# Discussion and Future Work

---

This thesis discussed the efforts towards fundamental understanding of the role of conformational switching in self-assembling systems, and application of conformational switching to design of self-assembling mechanical systems. This chapter provides the short summary of the work presented in this thesis, discusses some related issues, and states the contribution of the work. Finally, suggestion for future work is made.

## 6.1 Summary of the work

### **Parametric design optimization of simple mechanical conformational switches**

Parametric design optimization of two types of simple mechanical conformational switches, sliding bar mechanisms and minus devices was discussed. These mechanical conformational switches were used as building blocks of self-assembling parts for one-dimensional self-assembly via sequential random bin-picking. A genetic algorithm, in conjunction with computer simulation of sequential random bin-picking, optimized the parameterized switch designs to maximize the yield of a desired assembly. The results of genetic optimization are presented in the case of two, three and four part one-dimensional self-assembly. Rate equation analyses of the resulting designs revealed that sliding bar mechanisms could cause temporal change in part concentration by forming temporal intermediate assemblies,

and that minus devices could encode non-ambiguous and partially-ambiguous subassembly sequences. Effects of initial part concentration and defects during assembly are discussed. Principle of subassembly in biology is re-examined in the context of self-assembling mechanical systems. The following design guidelines, or “rules of thumb”, for  $n$ -part self-assembling systems are made based on these results:

- Non-ambiguous subassembly sequences yields better than ambiguous subassembly sequences.
- Parallel subassembly sequences yield better than linear subassembly sequences.
- Abundant parts should be assembled earlier rather than later.
- Parts with high defect probability should be assembled earlier rather than later.

### **Theory of one-dimensional self-assembling automata**

An abstract model of self-assembling systems is presented where assembly instructions are written as local rules that specify conformational changes of components. The model, self-assembling automaton, is defined as a sequential rule-based machine that operates on one-dimensional strings of symbols. An algorithm is provided for constructing a self-assembling automaton which self-assembles a one-dimensional string of distinct symbols in a given subassembly sequence. Classes of self-assembling automata are defined based on three classes of subassembly sequences described by assembly grammars. The minimum number of conformations is provided which is necessary to encode instances of each class of subassembly sequences. It is proven that the rules corresponding to the above two types of conformational switches with three conformations for each component are enough to encode any subassembly sequences of a string of distinct symbols with arbitrary length.

### **Design of micro “mouse trap”**

An implementation of conformational switch for micro assembly – a micro “mouse trap,” was presented. It has self-closing compliant latches that clamp a free planar part inserted between them. The self-closing of the latches is induced by the insertion of the part which releases the potential energy stored in the “cocked” state of the device. The self-closing



action of the micro mouse trap allows the inaccurate insertion of the free part (mouse) and leads to very accurate final positioning, with the insertion force required to induce the self-closing being very small.

## 6.2 Discussion

### 6.2.1 Bin-picking simulation vs. rate equation analyses

In Chapter 3, the space of parameterized conformational switch designs is searched to find the switch designs that maximize the yield of a desired assembly thorough computer simulation of sequential random bin-picking. And then, the resulting designs are analyzed using the rate equation to confirm, and often to get better understanding of the search results. This approach was chosen since designs found using the simulation may not be optimal in a statistical sense due to the stochastic nature of bin-picking simulation. Also, since it is impossible to run the simulation of each design for  $t \rightarrow \infty$ , a switch design is evaluated based on the “reaction rate” at the time of simulation termination, not the yield at  $t \rightarrow \infty$ . Rate equation analyses would give statistically reliable predictions of the behavior of parts during self-assembly for a long period of time. Hence a question arises: why use bin-picking simulation at the first place?

Using the bin-picking simulation, the space of switch design are directly searched. The size of the search space is  $2^{mn}$  where  $m$  is the number of bits used to encode the parameters for a part design, and  $n$  is the number of parts in the desired assembly. This grows exponentially as  $n$  increases. An alternative approach is using the rate equation to search over the space of subassembly sequences to find the optimal sequence, and then find the conformational switch designs that encode the subassembly sequence. In this case, the size of search space is  $\Omega(2^n)$ , where  $\Omega$  denotes an asymptotic lower bound <sup>1</sup>.

Even though the complexity of the problems are exponential in the both approaches, the first has several advantages over the second. First, by searching the space of parameterized switch designs directly, we find *only* the subassembly sequences which can be encoded by the given conformational switch model. Since some subassembly sequences cannot be

---

<sup>1</sup>The proof of this fact is found in Appendix E

realized by a conformational switch model, additional constraints must be added when searching over the space of subassembly sequences. It is difficult in general, however, to know exactly which subassembly sequences can be encoded by a given conformational switch model as discussed in the next section. Also, in the second approach, optimal switch designs corresponding to the optimal subassembly sequences must be generated, whereas in the first approach the optimal switch designs are found directly as a result of the search. The first approach also seems to fit naturally within the framework of genetic algorithms since there is a direct analogy between the short-order building blocks in the binary representation and the complementary bonding sites in the geometric representation of the conformational switches.

### 6.2.2 Limitation of the theory of one-dimensional self-assembling automata

The theory of one-dimensional automata presented in Section 4 is based on the abstraction of conformational switches as the local rules that specify conformational changes of a component. This abstraction made it possible to address the essential problems on the role of conformational switching in self-assembling systems, without dealing with unimportant issues associated with particular implementations of conformational switches.

Corollary 1, for example, gives a clear explanation why  $((A(BC))D)$  is un-encodable as discussed in Section 3.5.8. The conformational switches described in Section 3.2.2 use only minus devices, hence they are abstracted as a class I self-assembling automaton<sup>2</sup>. On the other hand,  $((A(BC))D)$  is in  $\text{SEQ}(\{p\}) \setminus L(G_I)$ <sup>3</sup>. According to Corollary 1, therefore,  $((A(BC))D)$  cannot be encoded using only minus devices. By using *both* a sliding bar mechanism *and* minus devices, it becomes possible to encode  $((A(BC))D)$  as shown in 3-60, since it is now a class II self-assembling automaton which can encode subassembly sequences in  $\text{SEQ}(\{p\}) \setminus L(G_I)$  by Theorem 4.

We must be careful, however, when applying these theorems to a particular implementation of conformational switches. For instance, it *cannot* be concluded from Theorem 7

---

<sup>2</sup>Recall attaching rules are abstraction of the function of minus devices.

<sup>3</sup>More precisely,  $((A(BC))D)$  is a basic subassembly sequences which is an instance of an assembly template in  $\text{SEQ}(\{p\}) \setminus L(G_I)$ .

that, for any subassembly sequence  $SEQ(\{p\}) \setminus L(G_{II})$ , one can design three-conformation conformational switches using both a sliding bar mechanism and minus devices, which encodes the subassembly sequence. This is because a sliding bar mechanism and a minus device cannot be implemented in a single “digit.” Due to this constraint, three-conformation switches such as the one shown in Figure 3-60 can encode much fewer subassembly sequences than their theoretical counterpart. Although the theorems in Chapter 4 can provide useful insights on the encoding power of conformational switches, it is difficult in general to determine which subassembly sequences can be encoded by a given conformational switch implementation because of the existence of such implementation-dependent constraints.

### 6.2.3 Potential applications of micro mouse trap

As listed in Section 5.2, the design of the micro mouse trap has several properties desired for micro assembly. In addition to a micro fastener in micro assembly, the unique properties of the micro mouse trap makes it suitable for a wide range of applications:

- **Assembly tools:** since the micro mouse trap is reusable, it can be used for temporal clamping/positioning of micro parts.
- **Connectors:** simple mechanical clamping of the micro mouse trap is suitable to electro-mechanical and opto-electrical connectors.
- **Actuators:** the induced fit property of the micro mouse trap can be applied to a micro reactive gripper.
- **Circuit breakers:** the easy integration with electrical components allows the micro mouse trap to be used as a micro circuit breaker.
- **Sensors:** with appropriate probing of the free part, the micro mouse trap can be a micro sensor for, *e.g.* force/impact, acceleration, displacement, pressure and temperature.

It is also of great interested in the use of the micro mouse trap in micro self-assembly processes such as those presented in [64],[11] and [29]. This would require re-designing of the energy releasing mechanism of the current micro mouse trap, such that it can be activated by a “feather touch” of free micro parts.

## 6.3 Contributions of this work

This work is the first attempt toward the fundamental understanding and application of conformational switching in self-assembling systems. The contributions of this work can be summarized as follows:

- **Parametric design optimization of simple mechanical conformational switches:** Design of two types of conformational switches, sliding bar mechanisms and minus devices, and identification of their roles in one-dimensional self-assembly processes through quantitative analyses. Quantitative discussion on robustness of non-ambiguous subassembly sequences in self-assembly processes. Design guidelines for  $n$ -part self-assembling systems based on quantitative analyses.
- **Theory of one-dimensional self-assembling automata:** Development of an abstract model of one-dimensional self-assembly, self-assembling automata. Mathematical proof of the fact that two types of conformational switches corresponding to sliding bar mechanisms and minus devices, can encode any subassembly sequences. Identification of classes of subassembly sequences and the corresponding classes of self-assembling automata with theoretical minimum conformations.
- **Design of micro “mouse trap”**  
Implementation of concept of conformational switching in micro-electromechanical systems (MEMS) and demonstration of the potential of conformational switching for assembly in MEMS.

## 6.4 Future work

### 6.4.1 Extensions to the theory of self-assembling automata

There are number of extensions which should be incorporated to the current theory of one-dimensional self-assembling automata. The current definition of classes of SA is based on the classes of subassembly sequences of one-dimensional string of *distinct* symbols, which we referred to as *basic subassembly sequences*. Many self-assembling systems in nature, however, often involves self-assembly of identical components. Therefore, the definition of

SA based on classes of *non-basic* subassembly sequences would be desirable. Also, most biochemical reactions are bi-directional; if the reaction  $a + b \rightarrow ab$  is possible, the reverse reaction  $ab \rightarrow a + b$  is also possible. Hence the current definition of SA should be extended such that the rule set can also contain detaching rules; rules of the form  $a^\alpha b^\beta \rightarrow a^\alpha + b^\beta$ . Accordingly, the definition of self-assembly must be modified. Finally, since the classification of SA presented in this paper is based on subassembly sequences, it can also be applied to the self-assembly in higher dimensions since. However, there are no concept of geometry and topology since the classes are developed for one-dimensional self-assembly. In order to extend SA to higher dimensions, these concepts must be incorporated.

#### 6.4.2 Turing completeness of one-dimensional self-assembling automata

In computational theoretical term, one-dimensional self-assembling automata (SA) discussed in Chapter 4 is a model of computation, and the theorems proven in the chapter identify the classes of languages accepted by one-dimensional SA. Under these view, a question arises immediately on the computational power of one-dimensional SA: if it can simulate every Turing machine. From its similarity with one-dimensional cellular automaton (CA), one can expect that one-dimensional SA is also Turing complete. In fact, it can be shown that one-dimensional SA is equivalent to one-dimensional three-neighbor CA with the  $H_1$  template described in [31]. Let us call this CA as one-dimensional  $H_1$  CA. Namely, for any one-dimensional  $H_1$  CA, there exists an one-dimensional SA that simulate it [48]. Since one-dimensional  $H_1$  CA is Turing complete [31], it can be concluded that one-dimensional SA is also Turing complete. Further, Smith [31] has also shown that there exists a 18-state one-dimensional  $H_1$  CA, and therefore there exists a one-dimensional SA with  $|Q| = 18$ , where  $Q$  is the conformation set of the SA. All these results are currently under documentation, and will be included in [48].

#### 6.4.3 Three part assembly using APOS

Along with the efforts of extending the theory to higher dimensions, I believe it is also very important to study some experimental systems which exhibits the conformational self-assembly in higher dimensions. Example of such mechanical systems are shown in Figure 6-1. It is a two-dimensional self-assembling system which consists of pie-shaped

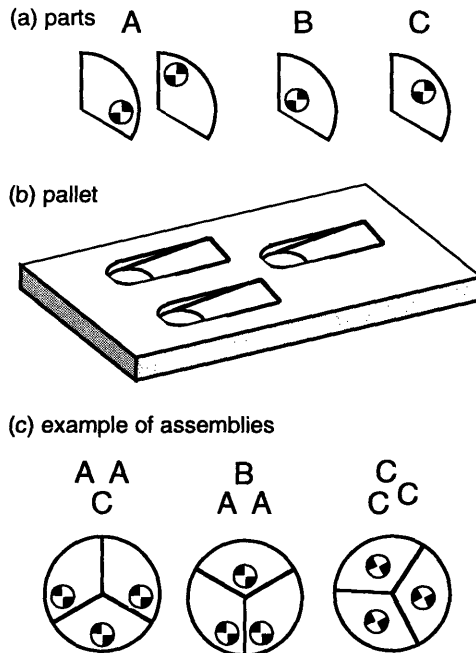


Figure 6-1: Three part self-assembly using APOS. (a) pie-shaped parts with biased locations of the center of gravity. (b) a pallet shape for the use with APOS. (c) examples of assemblies as a result of vibratory agitation: depending on the concentration of A, B and C, the some assembly would form with higher probability than the others.

components shown in Figure 6-1(a), which are assembled via vibratory agitation using SONY's Automated Parts Orienting System (APOS)<sup>4</sup>. Figure 2.1.1(b) shows an example of the pallet which can be used for the palletization of the pie-shaped parts into a disk. Each of these pie-shaped parts has an embedded weight which bias the location of their center of gravity. There are, for instance, three kinds of parts: A, B and C, depending of the locations of the center of gravity. Note in Figure 6-1(a), two parts shown as part A are identical since parts can flip during the vibratory agitation by APOS.

Due to the biased locations of the center of gravity, the palletization of the mixture of the pie-shaped parts A, B and C would results in the formation of three part assemblies (*i.e.* disks) with various orientations, such as the ones shown in Figure 2.1.1(c). Interest here is that how the the initial concentrations of parts A, B and C would affect the formation of a particular type of assembly. In particular, the concentrations which yields an assembly the most may not be the same as the part concentration in the assembly itself. One can

<sup>4</sup>See Section 2.1.1 for a short description of APOS.

expect, for example, an assembly shown in the leftmost figure in Figure 2.1.1(c) would have higher probability to form than the other when the initial concentration of part C is higher. The best yield of the assembly, however, would not be achieved when  $A:C = 2:1$  since this would result in many defects caused by the part A's trapped in the cavity in the wrong orientation *before* part C is palletized. Although no explicit conformational change takes place during the vibratory self-assembly of the assembly in the figure, part C has to be palletized *before* two part A's are palletized. The "conformational change" of the cavity occurs as a result of the integration with part C, which increases the probability of palletizing part A afterwards. This is analogous to the situation in three-dimensional biological self-assembly, where a conformational change which occurred at an assembly step provides the essential substrate for assembly at the next step [58]. Studying such experimental systems would be one of the most important steps beyond this thesis in the immediate future.

---




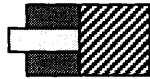



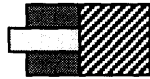







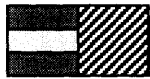
## Appendix A

# Optimal designs of two part non-randomized assembly

---

This appendix lists all the possible designs that score the maximum fitness (fitness = 2) in the two part sequential assembly discussed in Section 3.4.2. The length of the chromosome in this example is 14 (6 for part A and part B, and 2 for part Z), so there are  $2^{14} = 16384$  possible chromosomes. Since the assembly process is deterministic, depth first search can find all the optimal solutions with little enumeration. They are listed in Figures A-1 and A-2. Due to the degeneracy of parameter coding (see Section 3.4.1), more than one chromosome maps to a design. Also, some designs are functionally equivalent. Figure A-2 shows such designs equivalent to the two designs of part B appearing in Figure A-1. Since (0, F, 0, 0) has 4 equivalent part designs and (1, F, 0, 0) has 3 equivalent designs, there are  $4 \times 5 + 4 \times 4 = 36$  optimal designs.



Z	A	B	AB
 (0)	 (0, T, 1, 1)	 (0, F, 0, 0)	
 (0)	 (1, T, 1, 0)	 (0, F, 0, 0)	
 (1)	 (-1, T, 1, 1)	 (0, F, 0, 0)	
 (1)	 (0, T, 1, 0)	 (0, F, 0, 0)	

















Z	A	B	AB
 (0)	 (0, T, 1, 0)	 (1, F, 0, 0)	
 (0)	 (1, T, 1, -1)	 (1, F, 0, 0)	
 (1)	 (-1, T, 1, 0)	 (1, F, 0, 0)	
 (1)	 (0, T, 1, -1)	 (1, F, 0, 0)	

Figure A-1: Optimal designs for two-part sequential assembly.

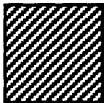

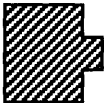

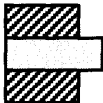
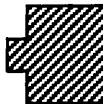


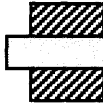
equivalent designs of part B				
				
(0, F, 0, 0)	(0, F, 0, -1)	(0, F, 0, 1)	(0, T, 0, 0)	(0, T, 1, 1)
(0, F, 1, 0)	(0, F, 1, -1)	(0, F, 1, 1)		
				
(1, F, 0, 0)	(1, F, 0, -1)	(1, F, 0, 1)	(1, T, 1, 1)	
(1, F, 1, 0)	(1, F, 1, -1)	(1, F, 1, 1)		

Figure A-2: Equivalent designs of part B.

---

## Appendix B

# Rate equation analysis of two part randomized assembly with a dummy part

---

This appendix explains details of rate equation analysis of two part randomized assembly with a dummy part discussed in Section 3.4.6. In the following, Design I and Design II refer to the part designs described in Section 3.4.6, unless otherwise specified. Derivation of the rate equations of these part designs follows the same steps found in Section 3.4.4.

The possible reactions for Design I are:



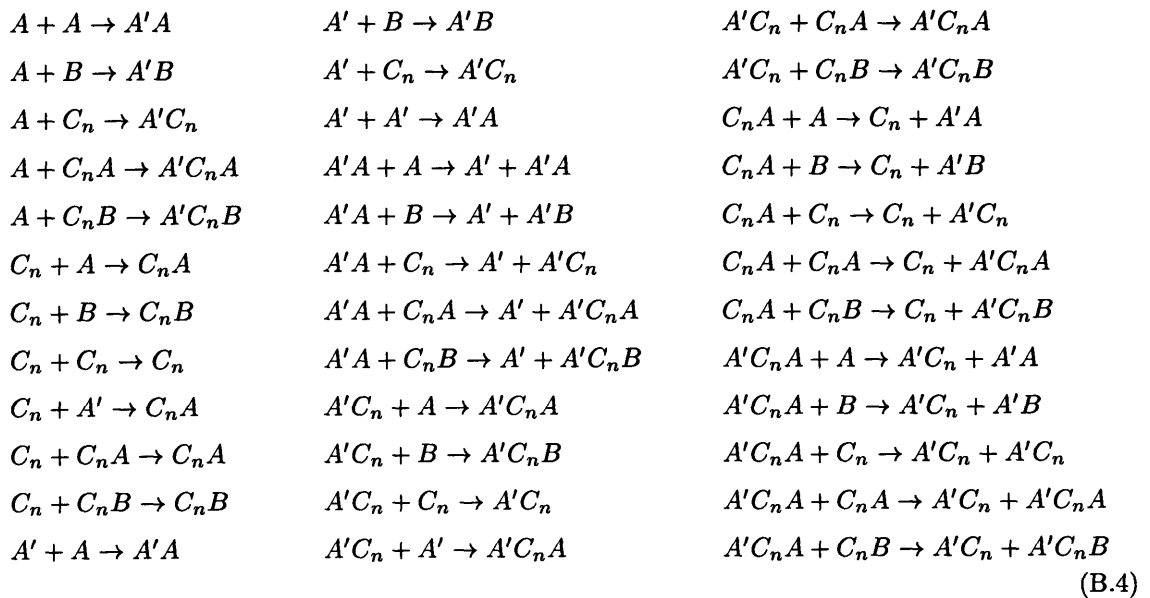
where  $A'$  denotes a part  $A$  after conformational change. And  $\mathbf{n}(t)$ ,  $\mathbf{A}$  and  $\mathbf{p}(t)$  are:

$$\mathbf{n}(t) = (n_A(t), n_B(t), n_C(t), n_{AA'}(t), n_{AB}(t))' \quad (\text{B.2})$$

$$A = \begin{pmatrix} -2 & -1 & 1 \\ 0 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{p}(t) = \begin{pmatrix} \frac{n_A(t) \cdot \{n_A(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \frac{n_{AA'}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \quad (\text{B.3})$$

where  $s(t) = n_A(t) + n_B(t) + n_C(t) + n_{AA'}(t) + n_{AB}(t)$ .

As easily seen in Figure 3-22, the number of possible reactions (and the number of possible subassemblies) can be very large for Design II since part  $C$ 's can form a subassembly  $C_n$  (an  $n$  concatenation of part  $C$ 's) up to 22 elements long. Fortunately,  $C$  is a solid part so there are no conformational changes in  $C$ : once a  $C$  binds to another  $C$ , the bond will never be destroyed. The resulting assembly  $CC$  will then behave *exactly* as a single  $C$ . This leads to the idea of not distinguishing part  $C_n$  and part  $C_m$  for any positive integer  $n$  and  $m$ . This reduces the number of subassemblies down to 11. The resulting rate equations are still useful since the desired assembly  $A'B$  does not contain part  $C$ 's. Therefore there is no need to keep track of the number of each  $C_n$ 's. Under the above assumption, there are 36 possible reactions of Design II:



(B.4)

where  $A'$  denotes a part  $A$  after conformational change, and  $C_n$  denotes a concatenation of *some* number of part  $C$ 's. Corresponding  $\mathbf{n}(t)$ ,  $A$  and  $\mathbf{p}(t)$  are, therefore, defined as follows:

$$\mathbf{n}(t) = (n_A(t), n_B(t), n_{C_n}(t), n_{A'}(t), n_{A'A}(t), n_{A'B}(t), n_{A'C_n}(t), n_{C_nA}(t), n_{C_nB}(t), n_{A'C_nA}(t), n_{A'C_nB}(t))' \quad (\text{B.5})$$

$$A = \begin{pmatrix} -2 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & -2 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 2 & 1 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & -1 & -2 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 & -1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{B.6})$$

$$\begin{aligned}
p_1(t) &= \frac{n_A(t) \cdot \{n_A(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} & p_2(t) &= \frac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} & p_3(t) &= \frac{n_A(t) \cdot n_{C_n}(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_4(t) &= \frac{n_A(t) \cdot n_{C_n A}(t)}{s(t) \cdot \{s(t) - 1\}} & p_5(t) &= \frac{n_A(t) \cdot n_{C_n B}(t)}{s(t) \cdot \{s(t) - 1\}} & p_6(t) &= \frac{n_{C_n}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_7(t) &= \frac{n_{C_n}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} & p_8(t) &= \frac{n_{C_n}(t) \cdot \{n_{C_n}(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} & p_9(t) &= \frac{n_{C_n}(t) \cdot n_{A'}(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{10}(t) &= \frac{n_{C_n}(t) \cdot n_{C_n A}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{11}(t) &= \frac{n_{C_n}(t) \cdot n_{C_n B}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{12}(t) &= \frac{n_{A'}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{13}(t) &= \frac{n_{A'}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} & p_{14}(t) &= \frac{n_{A'}(t) \cdot n_{C_n}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{15}(t) &= \frac{n_{A'}(t) \cdot \{n_{A'}(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} \\
p_{16}(t) &= \frac{n_{A' A}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} & p_{17}(t) &= \frac{n_{A' A}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} & p_{18}(t) &= \frac{n_{A' A}(t) \cdot n_{C_n}(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{19}(t) &= \frac{n_{A' A}(t) \cdot n_{C_n A}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{20}(t) &= \frac{n_{A' A}(t) \cdot n_{C_n B}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{21}(t) &= \frac{n_{A' C_n}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{22}(t) &= \frac{n_{A' C_n}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} & p_{23}(t) &= \frac{n_{A' C_n}(t) \cdot n_{C_n}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{24}(t) &= \frac{n_{A' C_n}(t) \cdot n_{A'}(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{25}(t) &= \frac{n_{A' C_n}(t) \cdot n_{C_n A}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{26}(t) &= \frac{n_{A' C_n}(t) \cdot n_{C_n B}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{27}(t) &= \frac{n_{C_n A}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{28}(t) &= \frac{n_{C_n A}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} & p_{29}(t) &= \frac{n_{C_n A}(t) \cdot n_{C_n}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{30}(t) &= \frac{n_{C_n A}(t) \cdot \{n_{C_n A}(t) - 1\}}{s(t) \cdot \{s(t) - 1\}} \\
p_{31}(t) &= \frac{n_{C_n A}(t) \cdot n_{C_n B}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{32}(t) &= \frac{n_{A' C_n A}(t) \cdot n_A(t)}{s(t) \cdot \{s(t) - 1\}} & p_{33}(t) &= \frac{n_{A' C_n A}(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\
p_{34}(t) &= \frac{n_{A' C_n A}(t) \cdot n_{C_n}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{35}(t) &= \frac{n_{A' C_n A}(t) \cdot n_{C_n A}(t)}{s(t) \cdot \{s(t) - 1\}} & p_{36}(t) &= \frac{n_{A' C_n A}(t) \cdot n_{C_n B}(t)}{s(t) \cdot \{s(t) - 1\}}
\end{aligned}
\tag{B.7}$$

For Design I and Design II, Equation 3.5 is solved numerically with the two initial conditions discussed in Section 3.4.6. Figure B-1 and Figure B-2 are the solutions for Design I and Design II with initial condition  $\mathbf{n}(0) = (40, 5, 5, 0, 0)'$ , and  $\mathbf{n}(0) = (40, 5, 5, 0, 0, 0, 0, 0, 0, 0)'$ , respectively. The same analysis is done with initial condition  $\mathbf{n}(0) = (22, 6, 22, 0, 0)'$  for

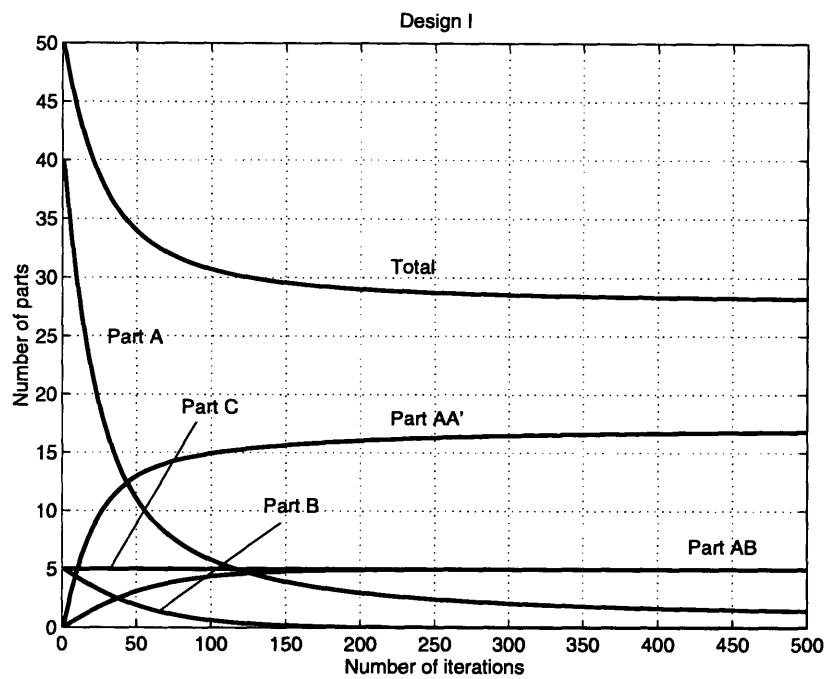


Figure B-1: Solution of equation 3.5 for Design I ( $A : B : C = 8 : 1 : 1$ ).

Design I, and  $\mathbf{n}(0) = (22, 6, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0)'$  for Design II. These results are shown in Figure B-3 and Figure B-4.

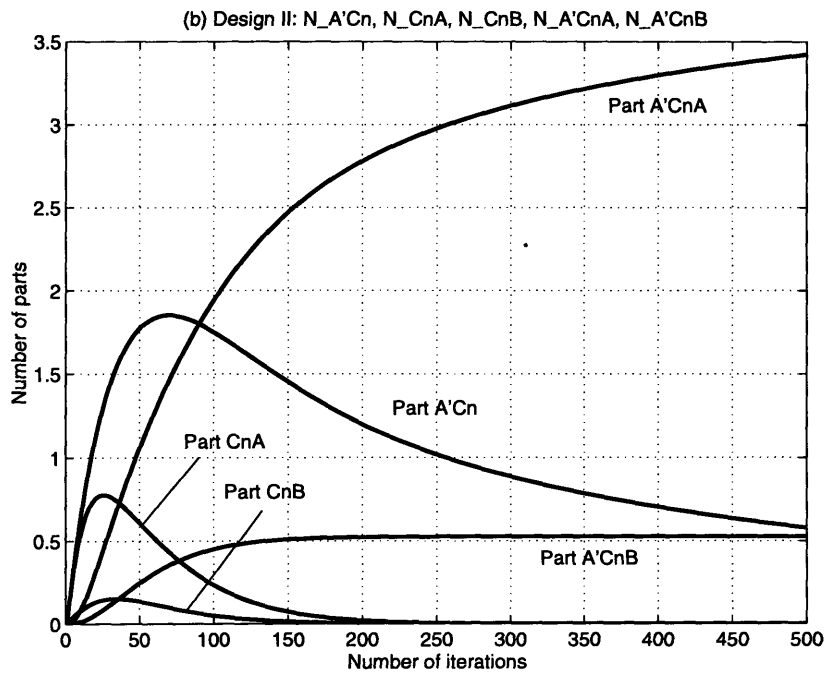
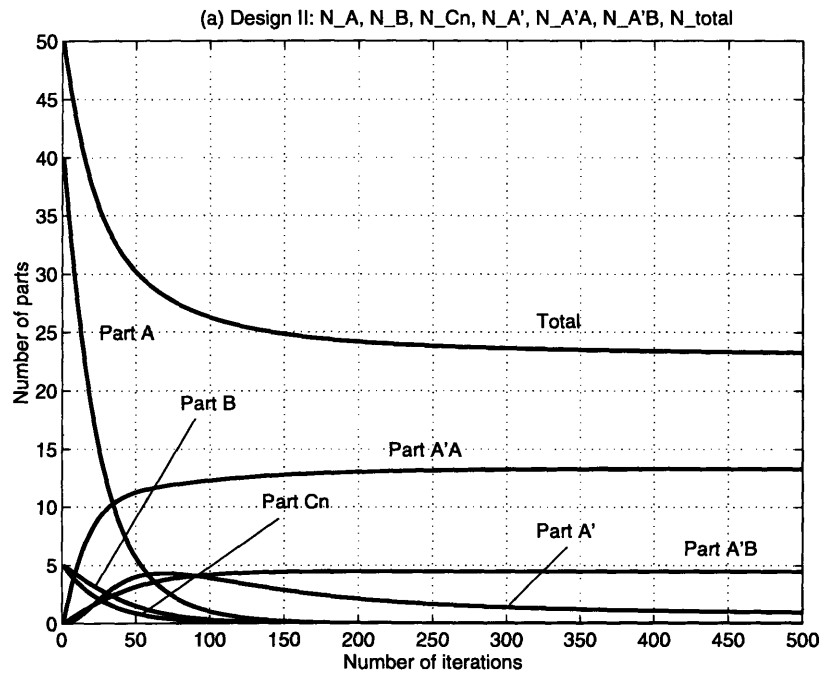


Figure B-2: Solution of equation 3.5 for Design II ( $A : B : C = 8 : 1 : 1$ ).



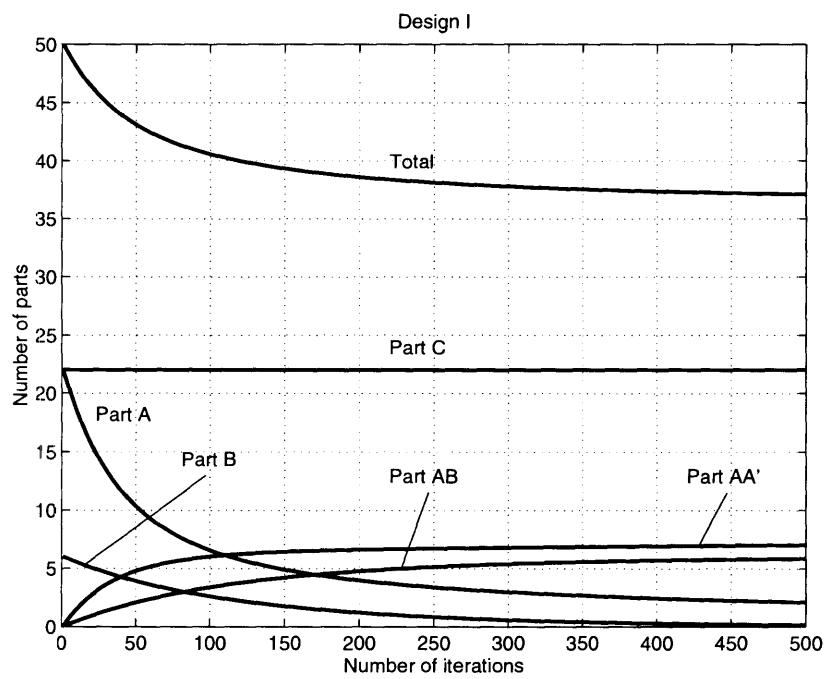


Figure B-3: Solution of equation 3.5 for Design I ( $A : B : C = 11 : 3 : 11$ ).

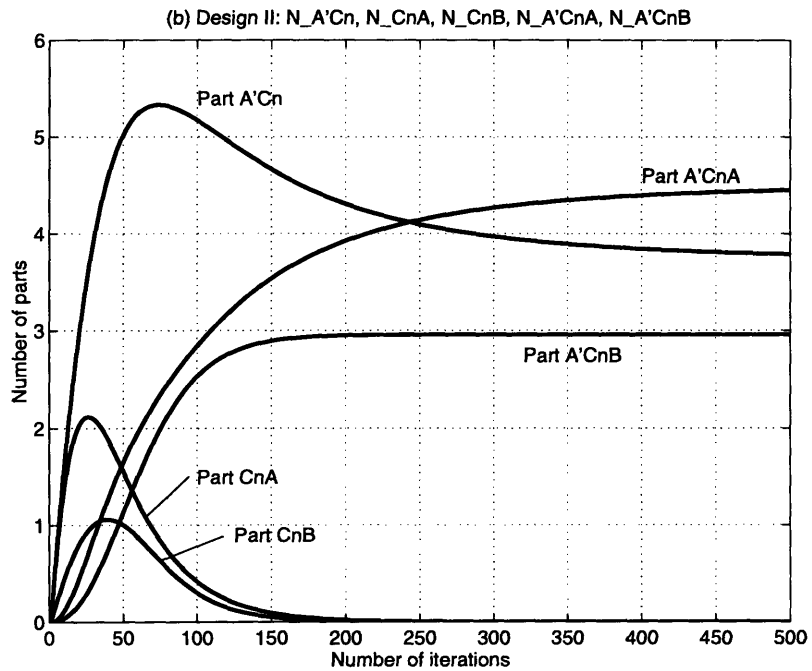
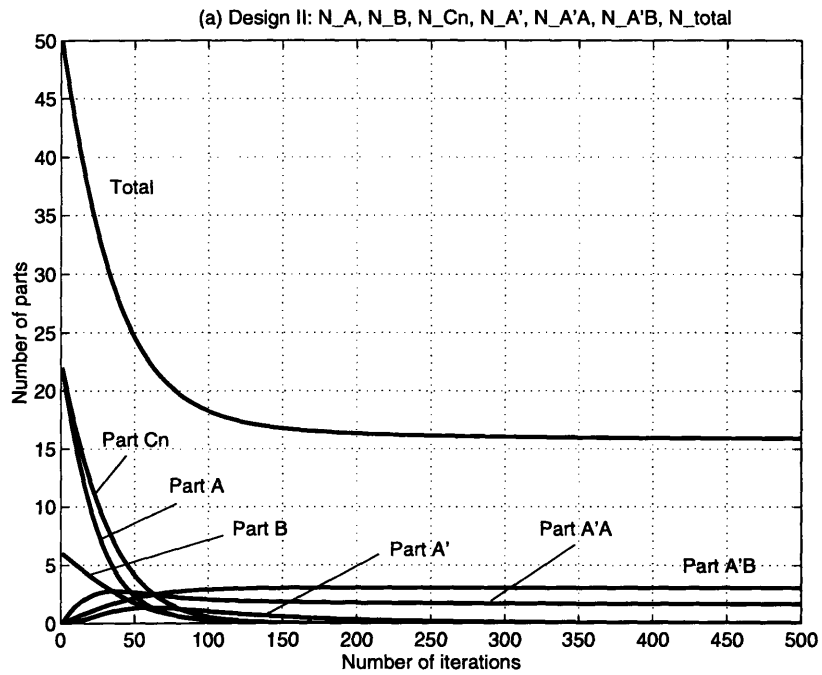


Figure B-4: Solution of equation 3.5 for Design II ( $A : B : C = 11 : 3 : 11$ ).

---

# Appendix C

## Rosen's subassembly model

---

This appendix outlines the derivation of Rosen's subassembly model, equations (2.1) and (2.2), described in Section 2.3.2.

Let  $N$  be the number of subassembly stages, and  $r_i$  be the number of subassemblies<sup>1</sup> produced at the  $(i - 1)$ -th assembly stage, which are incorporated into a single subassembly produced at the  $i$ -th assembly stage. Hence the number of elementary units in the final assembly  $L$  is:

$$L = r_1 \cdot r_2 \cdot \dots \cdot r_N \quad (\text{C.1})$$

We assume Crane's assumptions hold: 1) with probability  $q$  two subassemblies are put together wrongly, causing the resulting subassembly to be defective, and 2) the defective subassemblies cannot be incorporated into the subsequent subassemblies, so the elementary units in the defective subassemblies are completely wasted. These assumptions give the following recurrent expression of  $\nu_i$ , the number of non-defective subassemblies produced at the  $i$ -th assembly stage:

$$\begin{cases} \nu_i &= (1 - q)^{r_i} \left( \frac{\nu_{i-1}}{r_i} \right); & i \in \{1, 2, \dots, N\} \\ \nu_0 &= M \end{cases} \quad (\text{C.2})$$

---

<sup>1</sup>Note that  $r_1$  is the number of elementary units put together at the first assembly stage.

where  $M$  is the number of elementary units in the initial pool. The above recurrence is easily solved and yields  $\nu_N$ , the number of non-defective final assembly:

$$\nu_N = (1 - q)^{r_1 + r_2 + \dots + r_N} \left( \frac{M}{L} \right) \quad (\text{C.3})$$

Using the equation (C.3), we can calculate the yields of Crane's two subassembly processes described in Section 2.3.2. The yield of the first, three-stage subassembly process:

$$\nu_N = (1 - 0.01)^{10+10+10} \left( \frac{1,000,000}{1,000} \right) = 739 \quad (\text{C.4})$$

and for the second, one-stage subassembly process:

$$\nu_N = (1 - 0.01)^{1,000} \left( \frac{1,000,000}{1,000} \right) = 0.0432 \quad (\text{C.5})$$

Our objective is to choose the  $N + 1$  non-negative integers,  $N, r_1, r_2, \dots, r_N$ , such that the expression (C.3) is maximized, subject to the non-linear constraint (C.1). Equivalently in integer programming formulation (equations (2.1) and (2.2) in Section 2.3.2):

$$\begin{aligned} & \text{maximize} \quad (1 - q)^{r_1 + r_2 + \dots + r_N} \left( \frac{M}{L} \right) \\ & \text{subject to} \quad L = r_1 \cdot r_2 \cdot \dots \cdot r_N \\ & \quad \quad \quad N \geq 0; \quad N \in \mathbf{Z} \\ & \quad \quad \quad r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \dots, N\} \end{aligned}$$

By assuming different  $q$ 's at each subassembly stage,  $q_1, q_2, \dots, q_N$ , the solution of the recurrence (C.2) becomes:

$$\nu_N = (1 - q_1)^{r_1} (1 - q_2)^{r_2} \dots (1 - q_N)^{r_N} \left( \frac{M}{L} \right) \quad (\text{C.6})$$

and the corresponding integer programming formulation (equations (2.5) and (2.6) in Section 2.3.2) becomes:

$$\begin{aligned} & \text{maximize} \quad (1 - q_1)^{r_1} (1 - q_2)^{r_2} \dots (1 - q_N)^{r_N} \left( \frac{M}{L} \right) \\ & \text{subject to} \quad L = r_1 \cdot r_2 \cdot \dots \cdot r_N \end{aligned}$$

$$N \geq 0; \quad N \in \mathbf{Z}$$

$$r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \dots, N\}$$

which in general cannot be solved in closed form.

---

# Appendix D

## Proof of The Unique Factorization Theorem

---

**Theorem 8 (Unique Factorization Theorem)** *Let  $L$  be an positive integer, and  $L = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \dots \cdot p_k^{\alpha_k}$  be the unique factorization of  $L$  by the prime factors  $p_1, p_2, \dots, p_k$ . For any factorization of  $L = r_1 \cdot r_2 \cdot \dots \cdot r_m$ , the following inequality holds:*

$$\alpha_1 p_1 + \alpha_2 p_2 + \dots + \alpha_k p_k \leq r_1 + r_2 + \dots + r_m$$

*Proof:* Let  $l$  be a integer such that  $0 \leq l \leq k$ , and  $N = \{1, 2, \dots, k\}$ . Without loss of generality, an arbitrary factorization of  $L$  can be written as

$$L = p_1^{\alpha_1 - u_1} \cdot p_2^{\alpha_2 - u_2} \cdot \dots \cdot p_k^{\alpha_k - u_k} \cdot q_1 \cdot q_2 \cdot \dots \cdot q_l \tag{D.1}$$

where for  $i \in N$ ,  $0 \leq u_i \leq \alpha_i$ , and for  $j \in \{1, 2, \dots, l\}$ ,

$$q_j = \prod_{i \in N_j} p_i^{u_i} \tag{D.2}$$

and

$$N = \bigcup_{i=1}^l N_i; \quad N_i \cap N_j = \begin{cases} \emptyset & \text{if } i \neq j \\ N_i & \text{if } i = j \end{cases} \quad (\text{D.3})$$

Let  $S$  be the sum of the factors of this factorization, and  $S_0$  be the sum of the factors of the unique prime factorization. We wish to show that  $S_0 \leq S$  or equivalently  $S - S_0 \geq 0$ . Using the above notation,  $S$  can be written as

$$\begin{aligned} S &= (\alpha_1 - u_1)p_1 + (\alpha_2 - u_2)p_2 + \dots + (\alpha_k - u_k)p_k + q_1 + q_2 + \dots + q_l \\ &= (\alpha_1 p_1 + \alpha_2 p_2 + \dots + \alpha_k p_k) - (u_1 p_1 + u_2 p_2 + \dots + u_k p_k) \\ &\quad + q_1 + q_2 + \dots + q_l \end{aligned} \quad (\text{D.4})$$

Since  $S_0 = \alpha_1 p_1 + \alpha_2 p_2 + \dots + \alpha_k p_k$ ,

$$\begin{aligned} S - S_0 &= q_1 + q_2 + \dots + q_l - (u_1 p_1 + u_2 p_2 + \dots + u_k p_k) \\ &= \gamma_1 + \gamma_2 + \dots + \gamma_l \end{aligned} \quad (\text{D.5})$$

where for  $j \in \{1, 2, \dots, l\}$

$$\gamma_j = q_j - \sum_{i \in N_j} u_i p_i \quad (\text{D.6})$$

By Equation D.2,

$$\begin{aligned} \gamma_j &= q_j - \sum_{i \in N_j} u_i p_i \\ &= \prod_{i \in N_j} p_i^{u_i} - \sum_{i \in N_j} u_i p_i \\ &= \left( \prod_{i \in N_j} p_i^{n_j u_i} \right)^{\frac{1}{n_j}} - \frac{\sum_{i \in N_j} n_j u_i p_i}{n_j} \end{aligned} \quad (\text{D.7})$$

where  $n_j = |N_j|$ . We know for each  $j \in \{1, 2, \dots, l\}$ ,  $\gamma_j \geq 0$  since (geometric mean)  $\geq$  (arithmetic mean). By Equation D.5, therefore,  $S - S_0 \geq 0$ . ■

---

## Appendix E

# The size of the space of subassembly sequences

---

**Theorem 9** *The number of non-ambiguous subassembly sequences  $S_n$  of  $n$ -part one-dimensional assembly is  $S_n = \Omega(2^n)$ .*

*Proof:* Let  $k$  and  $l$  be positive integers such that  $k + l = n$ . Let us assume assembling  $n$  parts by assembling  $k$  parts and  $l$  parts separately, and then combining the  $k$ -part subassembly and  $l$ -part subassembly. In this particular case, the total number of non-ambiguous subassembly sequences of  $n$ -part assembly is  $S_k \cdot S_l$ . Since in the general case  $k$  and  $l$  can be any positive integer such that  $k + l = n$ ,  $S_n$  is the sum of all such cases:

$$\begin{aligned} S_n &= S_1 \cdot S_{n-1} + S_2 \cdot S_{n-2} + \dots + S_{n-1} \cdot S_1 \\ &= \sum_{i=1}^{n-1} S_i S_{n-i} \end{aligned} \tag{E.1}$$

where  $S_1 = 1$ . Let

$$T_n = \sum_{i=1}^{n-1} T_i \tag{E.2}$$

and  $T_1 = 1$ . Since  $\forall i \in \{1, 2, \dots, n\}$ ,  $S_i \geq 1$ , for any positive integer  $n$ ,

$$S_n \geq T_n \tag{E.3}$$



We know  $T_n = \Theta(2^n)$  since

$$\begin{aligned}T_{n-1} + T_{n-2} + \dots + T_1 &= (T_{n-2} + T_{n-3} + \dots + T_1) + T_{n-2} + \dots + T_1 \\&= 2(T_{n-2} + T_{n-3} + \dots + T_1) \\&= 2 \cdot 2(T_{n-3} + \dots + T_1) \\&\quad \vdots \\&= 2^{n-2}T_1 = 2^{n-2}\end{aligned}$$

where  $\Theta$  denotes an asymptotic tight bound. The equation (E.3) implies, therefore,  $S_n = \Omega(2^n)$ . ■

## Bibliography

- [1] J. Albert and K. Culik II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems*, 1:1–16, 1987.
- [2] G. K. Ananthasuresh and S. Kota. Designing compliant mechanisms. *Mechanical Engineering*, 117(11):93–96, 1995.
- [3] G. K. Ananthasuresh, S. Kota, and Y. Gianchandani. Systematic synthesis of micro-compliant mechanisms – preliminary results. In *The Third National Conference on Applied Mechanisms and Robotics Conference*, 1993.
- [4] G. K. Ananthasuresh, S. Kota, and N. Kikuchi. Strategies for systematic synthesis of compliant MEMS. In *ASME International Mechanical Engineering Congress and Exposition*, pages 677–686, 1994.
- [5] L. Y. Lin and S. S. Lee, M. C. Wu, and K. S. J. Pister. Micromachined integrated optics for free space interconnections. In *IEEE Micro Electro Mechanical Systems*, pages 77–82, 1995.
- [6] B. Berger, P. W. Shor, L. Tucker-Kellog, and J. King. Local rule-based theory of virus shell assembly. In *Proceedings of the National Academy of Science, USA*, pages 7732–7736, 1994. Vol. 91.
- [7] N. R. Branda, R. M. Grotzfeld, C. Valdés, and J. Rebek Jr. Control of self-assembly and reversible encapsulation of xenon in a self-assembling dimer by acid-base chemistry. *Journal of Americal .Chemical Society*, 117:85–88, 1995.
- [8] S. R. Burgett, K. S. J. Pister, and R. S. Fearing. Three dimensional structures made with microfabricated hinges. In *ASME International Mechanical Engineering Congress and Exposition*, pages 1–11, 1992.
- [9] S. Casjens. and J. King. Virus assembly. *Annual Review of Biochemistry*, 44:555–604, 1975.
- [10] D.L.D. Caspar. Switching in the self-control of self-assembly. In R. Markham and R. W. Horne, editors, *Structure-Function Relationship of Proteins*, pages 85–99, New York, NY, July 1976. North-Holland.
- [11] M. B. Cohn, C.-J. Kim, and A. P. Pisano. Self-assembling electrical networks: an application of micromachining technology. In *Transducers '91: 1991 Sixth International Conference on Solid-State Sensors and Actuators*, pages 490–493, New York, New York, 1991. IEEE.
- [12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press/McGraw-Hill, Cambridge, Massachusetts/New York, NY, 1989.

- [13] H. R. Crane. Principles and problems of biological growth. *The Scientific Monthly*, 70:376–389, 1950.
- [14] R. A. Crowther, E. V. Lenk, Y. Kikuchi, and J. King. Molecular reorganization in the hexagon to star transition of the baseplate of bacteriophage T4. *Journal of Molecular Biology*, 116:489–523, 1977.
- [15] R. Dizon, H. Han, and M. L. Reed. Single-mask processing of micromechanical piercing structures using ion milling. In *IEEE Micro Electro Mechanical Systems*, pages 48–52, 1993.
- [16] K. E. Drexler. *Nanosystems: molecular machinery, manufacturing and computation*. John Wiley & Sons, 1992.
- [17] M. R. Garey and D. S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- [18] N. S. Goel and R. L. Thompson. Movable finite automata (MFA): A new tool for computer modeling of living systems. In C. G. Langton, editor, *Artificial Life: the Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, pages 317–340, Los Alamos, New Mexico, September 1987. Addison Wesley.
- [19] N. S. Goel and R. L. Thompson. *Computer Simulations of Self-organization in Biological Systems*. Croom Helm, London, England, 1988.
- [20] N. S. Goel and R. L. Thompson. Movable finite automata (MFA) models for biological systems II: Protein biosynthesis. *Journal of Theoretical Biology*, 134:9–49, 1988.
- [21] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
- [22] R. M. Grotzfeld, N. R. Branda, and J. Rebek Jr. Reversible encapsulation of disc-shaped guests by a synthetic, self-assembled host. *Science*, 271:487–489, January 1996.
- [23] R. M. Grotzfeld, N. R. Branda, C. Valdés, and J. Rebek Jr. Control of self-assembly by acid-base chemistry. In J. S. Siegel, editor, *Proceeding of the NATO Advanced Research Workshop on Supermolecular Stereochemistry*, pages 195–197, Dordrecht, the Netherlands, 1994. Kluwer Academic Publisher.
- [24] H. Han, M. L. Reed, and L. E. Weiss. A mechanical surface adhesive using micromachined silicon structures. *Journal of Micromechanics and Microengineering*, 1(1):30–33, 1991.
- [25] H. Han, L. E. Weiss, and M. L. Reed. Mating and piercing micro-mechanical structures for surface bonding applications. In *IEEE Micro Electro Mechanical Systems*, pages 253–258, 1991.
- [26] H. Han, L. E. Weiss, and M. L. Reed. Micromechanical velcro. *Journal of Microelectromechanical Systems*, 1(1):37–43, 1992.
- [27] K. Hosokawa, I. Shimoyama, and H. Miura. Dynamics of self-assembling systems: Analogy with chemical kinetics. *Artificial Life*, 1(4):413–427, 1994.

- [28] K. Hosokawa, I. Shimoyama, and H. Miura. Dynamics of self-assembling systems: Analogy with chemical kinetics. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 172–180, Cambridge, Massachusetts, July 1994. The MIT Press.
- [29] K. Hosokawa, I. Shimoyama, and H. Miura. Two-dimensional micro-self-assembly using the surface tension of water. In *IEEE Micro Electro Mechanical Systems*, 1996. to appear.
- [30] K. Culik II, L. P. Hurd, and S. Yu. Computation theoretic aspects of cellular automata. *Physica D*, 45:357–378, 1990.
- [31] A. R. Smith III. Simple computation-universal cellular spaces. *Journal of the Association for Computing Machinery*, 18(3):339–353, July 1971.
- [32] M. W. Judy, Y.-H. Cho, R. T. Howe, and A. P. Pisano. Self-adjusting microstructures (SAMS). In *IEEE Micro Electro Mechanical Systems*, pages 51–56, 1991.
- [33] D. S. Lawrence, T. Jiang, and M. Levett. Self-assembling supermolecular complexes. *Chemical Reviews*, 95(6):2229–2260, 1995.
- [34] J.-M. Lehn. Supermolecular chemistry. *Science*, 260:1762–1763, June 1993.
- [35] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein. Quantum cellular automata. *Nanotechnology*, 4:49–57, 1993.
- [36] J. C. Martin. *Introduction to Language and the Theory of Computation*. McGraw-Hill, New York, New York, 1991.
- [37] M. L. Minsky. *Computation: finite and infinite machines*. Prentice Hall, 1967.
- [38] P. H. Moncevicz. Orientation and insertion of randomly presented parts using vibratory agitation. Master’s thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1991.
- [39] P. H. Moncevicz and M. J. Jakiela. Method and apparatus for automatic parts assembly. United States Patent 5,155,895, October 20 1992.
- [40] P. H. Moncevicz, M. J. Jakiela, and K. T. Ulrich. Orientation and insertion of randomly presented parts using vibratory agitation. In A. H. Soni, editor, *Proceedings of the ASME 3rd Conference on Flexible Assembly Systems*, pages 41–47, New York, NY, September 1991. The American Society of Mechanical Engineers. DE-Vol. 33.
- [41] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.
- [42] L. S. Penrose. Self-reproducing machines. *Scientific American*, 200:105–114, June 1959.
- [43] K. E. Peterson. Silicon as a mechanical material. *Proceedings of IEEE*, 70(5), 1982.
- [44] K. S. J. Pister, M. W. Judy, S. R. Burgett, and R. S. Fearing. Microfabricated hinges. *Sensors and Actuators A*, 33(3):249–256, 1992.

- [45] R. Prasad, K.-F. Bohringer, and N. C. MacDonald. Design, fabrication, and characterization of single crystal silicon latching snap fasteners for micro assembly. In *ASME International Mechanical Engineering Congress and Exposition*, 1995.
- [46] M. L. Reed, H. Han, and L. E. Weiss. Silicon micro-velcro. *Advanced Materials*, 4:48–51, 1992.
- [47] R. Rosen. Subunit and subassembly process. *Journal of Molecular Biology*, 28:415–422, 1970.
- [48] K. Saitou. On Turing completeness of one-dimensional self-assembling automata. *in preparation.*, 1996.
- [49] K. Saitou and M. J. Jakiela. Automated optimal design of mechanical conformational switches. *Artificial Life*, 2(2):129–156, 1995.
- [50] K. Saitou and M. J. Jakiela. Subassembly generation via mechanical conformational switches. *Artificial Life*, 2(4), 1995. to appear.
- [51] K. Saitou and M. J. Jakiela. Design of a self-closing compliant “mouse trap” for micro assembly. In *1996 International Mechanical Engineering Congress and Exposition (Winter Annual Conference of the American Society of Mechanical Engineers)*, November 1996. submitted for review.
- [52] K. Saitou and M. J. Jakiela. On classes of one-dimensional self-assembling automata. *Complex Systems*, 1996. submitted for review.
- [53] J. I. Steinfeld, J. S. Francisco, and W. L. Hase. *Chemical Kinetics and Dynamics*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.
- [54] R. L. Thompson and N. S. Goel. A simulation of T4 bacteriophage assembly and operation. *BioSystems*, 18:23–45, 1985.
- [55] R. L. Thompson and N. S. Goel. Movable finite automata (MFA) models for biological systems I: Bacteriophage assembly and operation. *Journal of Theoretical Biology*, 131:351–385, 1988.
- [56] W. Trimmer, P. Ling, C.-K. Chin, P. Orton, R. Gaugler, S. Hashmi, G. Hashmi, B. Brunett, and M. Reed. Injection of dna into plant and animal tissues with micromechanical piercing structures. In *IEEE Micro Electro Mechanical Systems*, pages 111–115, 1995.
- [57] L. C. Tucker. A local rule paradigm for the self-assembly of icosahedral viruses. Master’s thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, February 1993.
- [58] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner. *Molecular Biology of the Gene*. Benjamin/Cummings, Menlo Park, California, 1987.
- [59] G. M. Whitesides. Self-assembling materials. *Scientific American*, pages 146–149, September 1995.

- [60] G. M. Whitesides, J. P. Mathias, and C. T. Seto. Molecular self-assembly and nanochemistry: A chemical strategy for the synthesis of nanostructures. *Science*, 254:1312–1319, November 1991.
- [61] S. Wolfram. Computation theory of cellular automata. *Mathematical Physics*, 96:15–57, 1984.
- [62] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.
- [63] H. J. Yeh and J. S. Smith. Fluidic self-assembly of microstructures and its application to integration of GaAs on Si. In *IEEE Micro Electro Mechanical Systems*, pages 279–284, New York, New York, 1991. IEEE.
- [64] H. J. Yeh and J. S. Smith. Fluidic self-assembly of GaAs microstructures on Si substrates. *Sensors and Materials*, 6(6):319–332, 1994.