On the Hierarchical Construction of *SL* Blocks

A Generative System that Builds Self-Interlocking Structures

Shen-Guan Shih

S. Shih
Department of Architecture, National Taiwan University of Science and Technology, Taiwan
sgshih@mail.ntust.edu.tw

S. Adriaenssens, F. Gramazio, M. Kohler, A. Menges, M. Pauly (eds.): Advances in Architectural Geometry 2016
© 2016 vdf Hochschulverlag AG an der ETH Zürich, DOI 10.3218/3778-4_10, ISBN 978-3-7281-3778-4
http://vdf.ch/advances-in-architectural-geometry-2016.html

Abstract

This paper explores the potential of forming hierarchical structures using just one type of element, called SL block. SL block is an octocube composed of an S-shaped and an L-shaped tetracubes attaching to each other side by side. SL blocks can be systematically assembled into variations of interlocking structures called SL strands. Multiple SL strands can be used as basic elements to build larger and stronger structures. A generative process of SL strands based on syntax-directed translation of high-level geometric specifications is defined to formalise the analysis and synthesis of forms that can be constructed with interlocking SL blocks. With the system it is not difficult to design forms that can be built by SL blocks in a top-down manner. SL blocks can be assembled to form large and firm structures without using mortise/tenon, glue, or nail. The construction can be repetitively dissembled and reassembled into various forms. The assembly process can be guided with sequential instructions so that very sophisticated structures can be encoded into compact and efficient specifications for construction.

Keywords:
interlock, polycube, generative system

1. Introduction

The research described in this paper uncovers a specific type of polycube, called *SL* block, which is an octocube consisting of an S-shaped and an L-shaped tetracubes attaching to each other side by side as shown in Figure 1. *SL* blocks can be used to assemble extendable self-interlocking structures. Large and stable structures can be constructed with thousands of *SL* blocks without using mortar, glue, or any adhesive materials. A set of generative rules of building interlocking structures with *SL* blocks was discovered. A generative system is proposed to enable systematic methods for the design and assembly of composite structures.

Interlocking is an interesting issue that is very useful in timber and prefabricated constructions. Advances of digital fabrication technology drive researches towards automatic generation of interlocking parts for assembly (Song et al. 2012).

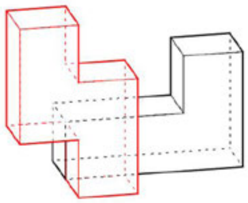

Figure 1. An SL block consists of an S-shaped and an L shaped tetracubes.

Interlocking levels of assembled structures can be distinguished by calculating the degrees of translational freedom for individual parts of the structure as well as the network of relations for parts engagements (Fu et al. 2015). Among these researches, polycubes were often used as the basic elements (Lo et al. 2009; Song et al. 2012; Song et al. 2015).

The discussion is further extended to uncover the top-down design method of constructions with higher levels of hierarchy based on interlocking *SL* blocks. The generative mechanism is defined with context-free string grammars, which is fundamentally different from the shape grammar devised by Stiny (1980). Shape grammar is based on the processing of non-monotonic shapes, which are regarded as dividable constructs that allow non-deterministic recognition and processing of shape features. Shape grammar is inevitably coupled with the problem of being non-computable for which its grammar rules, with the required non-terminal shapes in the left-hand sides, are all context sensitive. For shape generative methods, Shih (1994) took a different approach by using string grammars to generate sequences of symbols that specify shape creation processes. String grammars have been successfully used for the compilation of high-level programming languages since the 1970s. Well-developed methods based on string grammars have been proven to be efficient and effective for the analysis and synthesis of syntactic structures that can be defined with context-free grammars.

2. Engagements, Strings and Strands of *SL* Blocks

The engagement of two *SL* blocks is defined with the transformation that transforms one *SL* block to the other block that is attaching to it. Figure 2 shows 10 types of engagements for *SL* blocks that form the basic structure of interlocking configurations. The blue (darker) one in each figure represents the host block, which receives a grey (lighter) block as the guest for the engagement. A geometric transformation can be defined to transform the blue to the gray. Each engagement is named with an upper case letter if the engaging position is at the L part of the host block, and is named with a lower case letter if the engagement takes place at the S part of the host block.

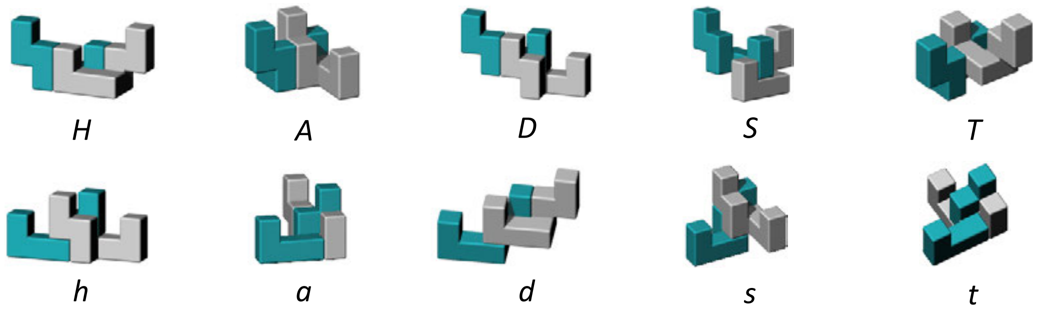

Figure 2. 10 types of engagements for SL blocks.

A string of engagements specifies the construction process of an *SL* string by starting with an initial block and adding on more with sequential applications of engagements in the string. For example, the string *HhH* specifies a string of four *SL* blocks lining up to form the configuration shown in the left-hand side of Figure 3. Respectively, *SL* strings of *aaa* and *ddd* are shown in the center and the right-hand side of Figure 3.

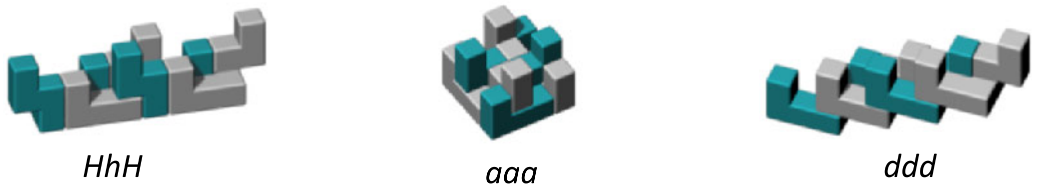

Figure 3. SL strings represented with corresponding types of engagements.

Engagements with upper and lower cases of the same letter are conjugates to each other. For every *SL* string, a conjugating string can be derived by replacing each engagement with its conjugate. Two conjugating strings of *SL* blocks can be placed against each other from the tops to form an interlocking structure called an *SL* strand. Figure 4 shows three simple interlocking strands formed by placing conjugating strings against the three *SL* strings shown in Figure 3. In this paper we use # as the notation for the operation that combines two conjugates together to form a strand. Since it is trivial to derive the conjugate of an *SL* string, the second string in an *SL* strand can be omitted in the notation.

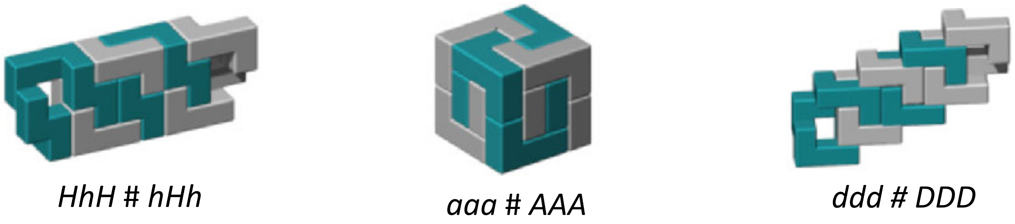

Figure 4. SL strands with conjugating strings of three engagements.

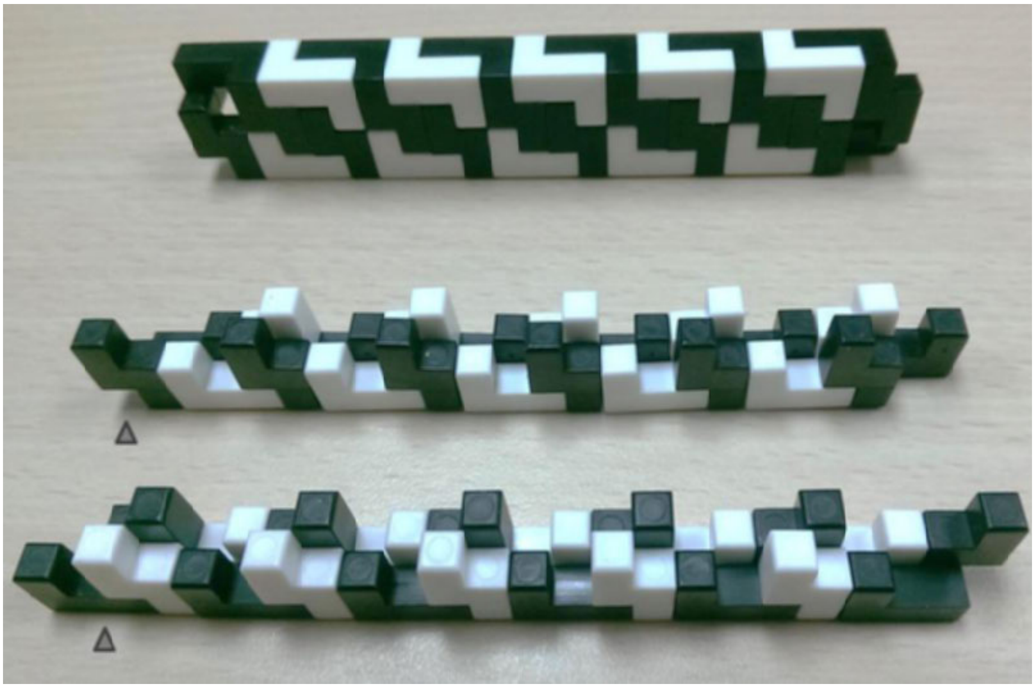

Figure 5. The conjugating strings of HhHhHhHhHhH (center) and hHhHhHhHhHh (bottom). The interlocking strand built with the two strings (top).

3. Interlocking

The interlocking mechanism of elements can be distinguished by whether it is locked by topology or by friction. With topological interlocking (Dyskin 2003), the whole structure would not be broken until some elements have already been broken. If the element is locked by friction, then the structure falls apart when the dragging force overpowers the friction. The interlocking structure of *SL* strands is not totally topological. Indeed, if any composite structure with parts that are jointed totally with topological locking, there would be no way to disassemble the structure without breaking some of the parts. Take the long strand formed by *HhHhHhHhHhH* and its conjugate (Fig. 5) as an example. Interlocking is topological when applying forces along the axial direction of the strand. The strand will

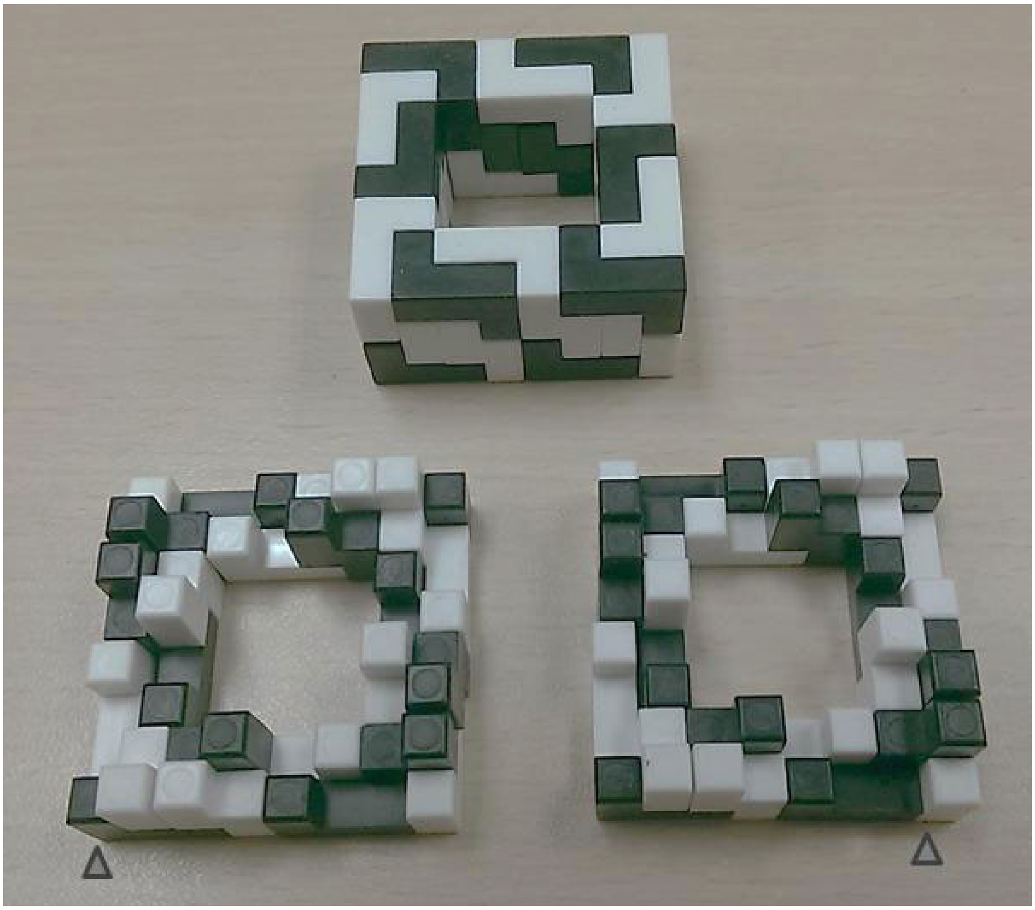

Figure 6. The conjugating strings of HhaHhaHhaHh(a) (bottom left) and hHAhHAhHAhH(A) (bottom right). The interlocking strand built with the two strings (top).

not be broken unless it is pulled apart along its axis hard enough to break some of the blocks. When the dragging force is applied vertically, which is along the same direction as the conjugate string is pushed into the host string, the locking is held only by friction. Even so, except for the two blocks that are located at the very ends of the strand, no single *SL* block can be pulled out of the strand without taking with it at least one other block. The weak points of the ends disappear when the strand joints its two ends to form a cyclic configuration such as the strand of *HhAHhAHhAHA(A) # hHahHahHahH(a)* (Fig. 6). The engagement enclosed by parenthesis is for the emerging engagement that takes place when the last *SL* block engages with the first.

4. Syntax-Directed Translation of *SL* Strings

Syntax-directed translation is a method of compiler implementation for translating the input code in one language to its corresponding code in another language. The definition of a specific syntax-directed translation requires an input grammar, which is used to derive the syntactic structure of the input, and an output grammar, which is used to generate the output. Our purpose is to derive the construction process of *SL* strands for a given shape.

With the notation we used, the plus sign represents options for what is to substitute with, and 1 represent the null string. The translation is defined as such:

Non-terminal		Production rules for input string	Production rules for output string
Init: $\boldsymbol{X}_T$			
$\boldsymbol{X}_T \rightarrow$	1.	$\boldsymbol{X}_S$	$\boldsymbol{X}_S$
	2.	$\boldsymbol{X}_L$	$\boldsymbol{X}_L$
	3.	1	1
$\boldsymbol{X}_L \rightarrow$	4.	$ff\ \boldsymbol{X}_S$	$H\ \boldsymbol{X}_S$
	5.	$frf\ \boldsymbol{X}_L$	$A\ \boldsymbol{X}_L$
	6.	$dff\ \boldsymbol{X}_L + fdf\ \boldsymbol{X}_L + ffd\ \boldsymbol{X}_L$	$D\ \boldsymbol{X}_L$
	7.	$ufrf\ \boldsymbol{X}_S + furf\ \boldsymbol{X}_S + fruf\ \boldsymbol{X}_S + frfu\ \boldsymbol{X}_S$	$T\ \boldsymbol{X}_S$
	8.	$dflf\ \boldsymbol{X}_S + fdlf\ \boldsymbol{X}_S + frdf\ \boldsymbol{X}_S + flfu\ \boldsymbol{X}_S$	$S\ \boldsymbol{X}_S$
	9.	1	1
$\boldsymbol{X}_S \rightarrow$	10.	$ff\ \boldsymbol{X}_L$	$h\ \boldsymbol{X}_L$
	11.	$flf\ \boldsymbol{X}_S$	$a\ \boldsymbol{X}_S$
	12.	$uff\ \boldsymbol{X}_S + fuf\ \boldsymbol{X}_S + ffu\ \boldsymbol{X}_S$	$d\ \boldsymbol{X}_S$
	13.	$dflf\ \boldsymbol{X}_L + fdlf\ \boldsymbol{X}_L + fldf\ \boldsymbol{X}_L + flfd\ \boldsymbol{X}_L$	$t\ \boldsymbol{X}_L$
	14.	$ufrf\ \boldsymbol{X}_S + furf\ \boldsymbol{X}_S + fruf\ \boldsymbol{X}_S + frfu\ \boldsymbol{X}_S$	$s\ \boldsymbol{X}_S$
	15.	1	1

The input grammar defines a language that uses letters to represent the path of the *SL* string. The definitions of letters are listed as follows:

f: move one step forward
r: turn right for 90 degrees
l: turn left for 90 degrees
u: move one step up
d: move one step down

With this representation, a straight line of n steps can be written as f^n. A square with 6 steps in width can be written as $f^5rf^6rf^6rf^6rf$. With the above syntax-directed translation, the square would be translated into the *SL* string $(HhA)^4$. With the string, it is trivial to derive the conjugate to build the corresponding strand, as the photo shows in the middle of Figure 6. The deviation process of the output string is listed as follows:

Input : Output

Rule	Input	Output
$\boldsymbol{X}_L$	$\boldsymbol{X}_L$	
4.	$ff\ \boldsymbol{X}_S$	$H\ \boldsymbol{X}_S$
10.	$f^4\ \boldsymbol{X}_L$	$Hh\ \boldsymbol{X}_L$
5.	$f^5rf\ \boldsymbol{X}_L$	$HhA\ \boldsymbol{X}_L$
4.	$f^5rf^3\ \boldsymbol{X}_S$	$HhAH\ \boldsymbol{X}_S$
10.	$f^5rf^5\ \boldsymbol{X}_L$	$HhAHh\ \boldsymbol{X}_L$
5.	$f^5rf^6rf\ \boldsymbol{X}_L$	$HhAHhA\ \boldsymbol{X}_L$
4.	$f^5rf^6rf^3\ \boldsymbol{X}_S$	$HhAHhAH\ \boldsymbol{X}_S$
10.	$f^5rf^6rf^5\ \boldsymbol{X}_L$	$HhAHhAHh\ \boldsymbol{X}_L$
5.	$f^5rf^6rf^6rf\ \boldsymbol{X}_L$	$HhAHhAHhA\ \boldsymbol{X}_L$
4.	$f^5rf^6rf^6rf^3\ \boldsymbol{X}_S$	$HhAHhAHhAH\ \boldsymbol{X}_S$
10.	$f^5rf^6rf^6rf^5\ \boldsymbol{X}_L$	$HhAHhAHhAHh\ \boldsymbol{X}_L$
5.	$f^5rf^6rf^6rf^6rf\ \boldsymbol{X}_L$	$HhAHhAHhAHhA\ \boldsymbol{X}_L$
9.	$f^5rf^6rf^6rf^6rf$	$HhAHhAHhAHhA$

Syntax-directed translation (Aho et al. 1986) can be used to implement tools for automatic generation of *SL* strand based on figures drawn by the user. With the parser for the input grammar, the process can also check if it is possible to create an *SL* strand for the desirable form. The tool can use the input grammar to guide the form-creation process so that only forms that can be assembled with *SL* blocks would be drawn. Even when erroneous forms are given as input and the parsing fails, some syntax-directed methods such as error-correcting parsing

(Aho et al. 1972) might be able to adapt the input form so that some similar shapes that are buildable can be created. With the assistance of such tools, it would not be difficult to create more sophisticated structures using multiple *SL* strands.

5. Strand Hierarchy

Interlocking can be implemented at various levels, among which the lowest is built upon the engagements of *SL* blocks. Since *SL* strands are stable constructions formed by interlocking *SL* blocks, they can be used as basic elements to create super structures built upon multiple *SL* strands. The process may go on recursively for higher hierarchy still. Figure 7 shows a structure built with eight interlocking rectangular rings arranged like a cyclic chain. The ring-like strands are moveable but cannot be totally separated from each other.

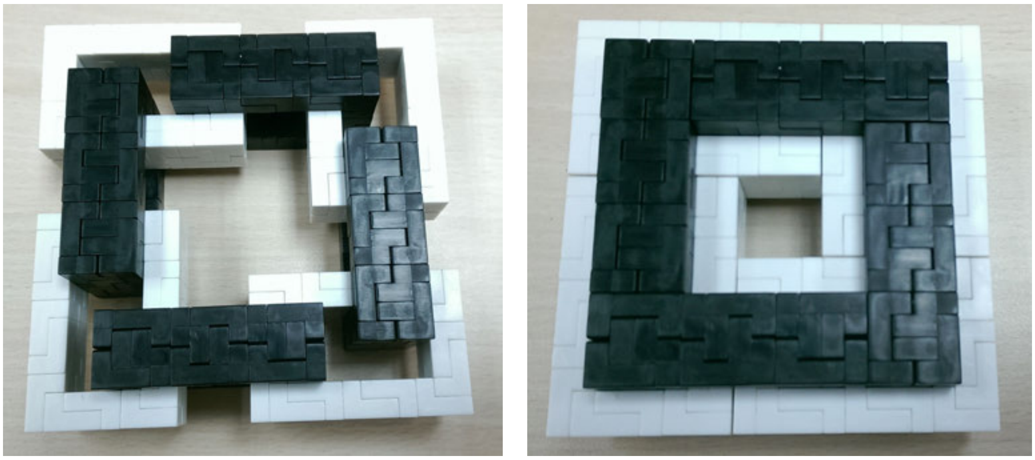

Figure 7. A chain structure with 8 moveable parts built with SL strands of (ahHhHahH)^2.
left: expanded; right: contracted.

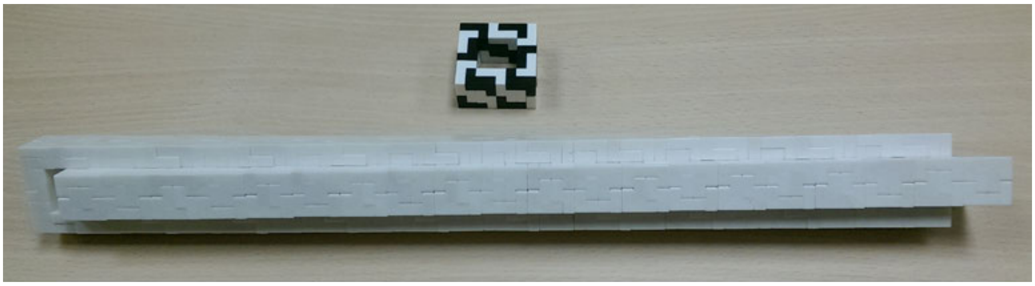

Figure 8. A chain structure with non-moveable parts built with 20 SL strands.

Figure 8 shows a long stick consisting of 480 *SL* blocks. The structure is formed by 20 interlocking square rings identical to the one that is shown at the top of the figure. Except for the two ends, each internal ring is locked by two neighboring rings that are oriented perpendicularly. The structure effectively prevents individual strand from breaking when forces are applied to twist the entire chain.

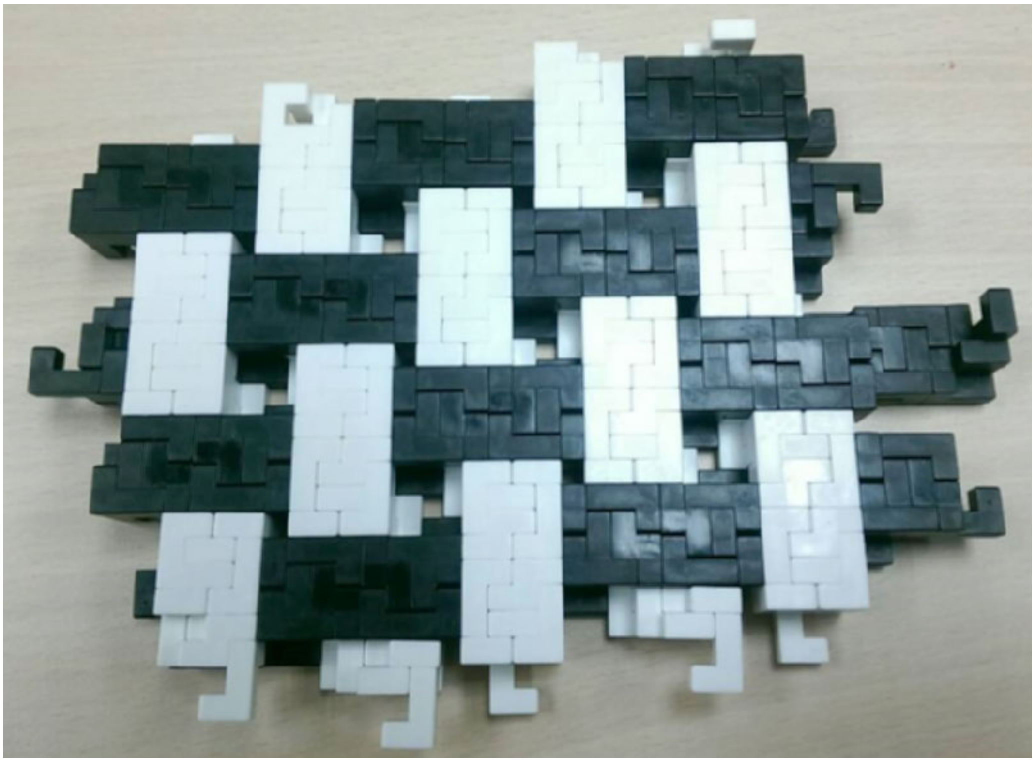
Figure 9a. A structure with interlaced strands.

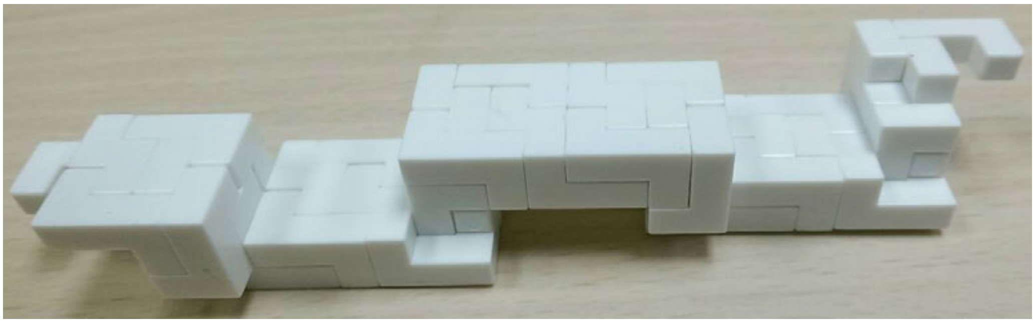
Figure 9b. One of the strands for the interlaced structure.

Figure 9(a) shows a structure built with 9 interlaced *SL* strands like the one shown in Figure 9(b), which can be represented as *(asHhTahH)n#(AShHtAHh)*^{n}. Interlacing could be an effective means to substentially extend the size and strength of the structure for practical applications. Each strand is strenthened by perpendicular strands that are locked with it. Except for the *SL* blocks that are located at the boundary, all internal blocks are topologically interlocked by others.

Figure 10(a) shows three structures composed of spiral strands. The one on the left consists of just one spiral strand, represented as *(St)n#(sT)*^{n}. The center one consists of two interlocked spiral strands, each of which is represented as *(add)*^{4n}*#(ADD)*^{4n}. The structure on the right consists of four spiral strands interlocked to form an integrity. Each of the strand is represented as (adddd)^{4n}#(ADDDD)^{4n}. (Fig. 10 b)

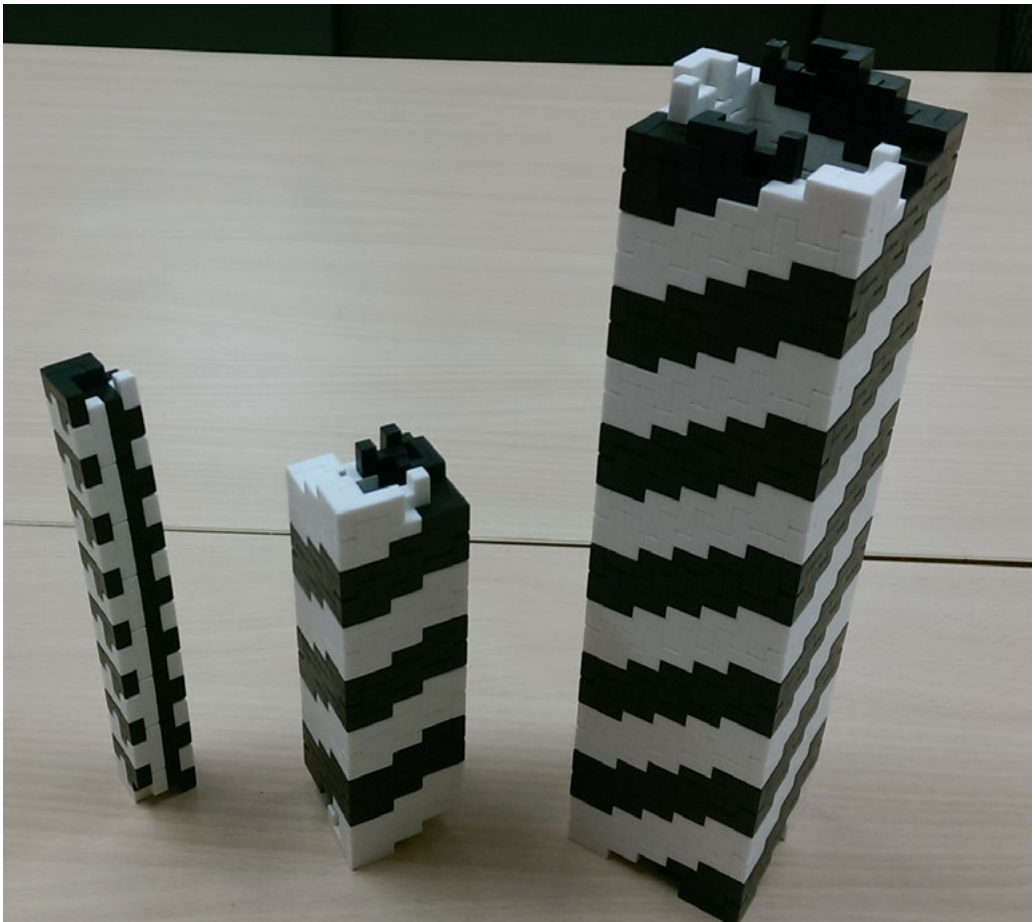

Figure 10a. Three structures with spiral strands.

6. Conclusion

SL block is a simple element with great potential for building larger composite structures. It is feasible to create hierarchical structures using multiple *SL* strands. Generative rules for the engagements of *SL* blocks enable efficient means for the analysis and synthesis of forms that could be built with *SL* blocks. Its usage for architecture is yet to be discovered. Various materials, details, sizes of *SL* blocks, etc., should be tested for practical applications.

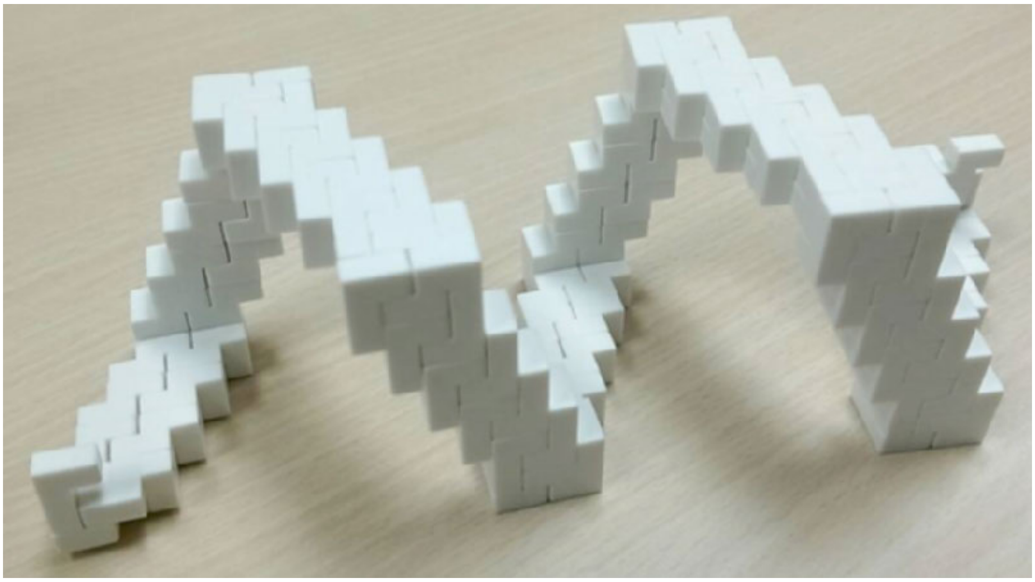

Figure 10b. The spiral strand of (adddd) 4n.

Acknowledgements

The research described in this paper is support by the Ministry of Science and Technology, Taiwan, R.O.C.

References

Aho, Alfred V., Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison Wesley.

Aho, Alfred V., and Thomas G. Peterson. 1972. "A Minimum Distance Error-Correcting Parser for Context-Free Languages." *SIAM Journal on Computing* 1, 4: 305–312. doi: 10.1137/0201022

Dyskin, A.V., Y. Estrin, A.J. Kanel-Belov, and E. Pasternak. 2003. "Topological Interlocking of Platonic Solids: A Way to New Materials and Structures." *Phil Mag Letters* 83, 3: 197–203.

Fu, Chi-Wing, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. 2015. "Computational Interlocking Furniture Assembly." *ACM Transactions on Graphics* 34, 4: Article 91, doi: 10.1145/2766892

Lo, Kui-Yip, Chi-Wing Fu, and Hongwei Li. 2009. "3D Polyomino Puzzle." *ACM Trans. Graph*. 28, 5: Article 157 doi: 10.1145/1618452.1618503

Shih, Shen-Guan. 1994. "The Use of String Grammars in Computer Aided Architectural Design." Doctoral thesis, Swiss Federal Institute of Technology.

Song, Peng, Chi-Wing Fu, and Daniel Cohen-Or. 2012. "Recursive Interlocking Puzzles." *ACM Transactions on Graphics*, 31, 6: Article 128, doi: 10.1145/2366145.2366147

Song, Peng, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. 2015. "Printing 3D Objects with Interlocking Parts." *Computer Aided Geometric Design* 35–36: 137-148. doi: 10.1016/j.cagd.2015.03.020

Stiny, George. 1980. "Introduction to Shape and Shape Grammars." *Environment and Planning B* 7: 343–351.