Grammars of *SL* Block Construction

Shen-Guan Shih

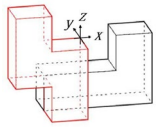
National Taiwan University of Science and Technology; sgshih@mail.ntust.edu.tw

Abstract

An ***SL*** block is an octocube that can be used to build semi-interlocking structures called ***SL*** strands. The study uses context-free grammars to define languages of ***SL*** strands. Several grammars are devised to display the relation between string rewrite rules and ***SL*** block constructions. It is expected that with syntactic operations based on grammars, computer modeling tools can be developed to assist artistic creations by enabling transformations and form finding between high level abstractions and low level constructions.

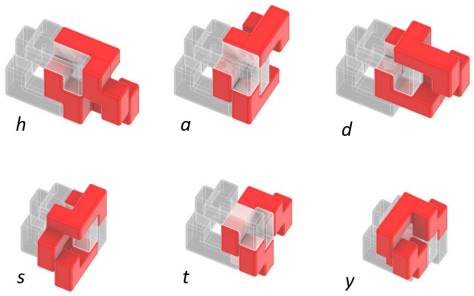
SL block, conjugate pair, and strand

An ***SL*** block is a kind of octocube that can be used to build semi-interlocking structures, for which most blocks are locked topologically while allowing some parts being held by friction and left with just one direction of translational freedom [1][2]. A semi-interlocked structure may retain stability under forces from various directions, and yet allows at least one feasible sequence for assembling and disassembling. Two ***SL*** blocks arranged into 180 degree ***Y*** axis rotational symmetry are called a conjugate pair, as shown in Figure 1(b). Conjugate pairs of ***SL*** blocks can be sequentially concatenated to build a kind of linear structure called ***SL*** strand. Six types of concatenations are denoted *h, a, d, s, t,* and *y,* with each of which the concatenated strand can be appended onto a specific direction and position. The ***SL*** strands may form enclosed loops if both ends meet at the same location. All looping ***SL*** strands are semi-interlocking.



*(a) an **SL** block*

(b) a conjugate pair



(c) six types of concatenations

Figure 1: ***SL*** *block, conjugate pair, and concatenations*

Grammars with rewrite rules on *SL* strands

The ***SL*** strands can be represented as strings consisting of six letters that stand for the six ways of concatenations. A grammar consists of an initial non-terminal and a set of rewrite rules, which are used to transform the initial non-terminal into strings consisting of only terminals, which are *h, a, d, s, t,* and *y*. In this paper, all non-terminals in grammar definitions are written as bold capital letters. The letter *e* represents empty concatenation that ends the assembly when the strand is not looped. The grammar $\boldsymbol{G_1}$ defines all possible ***SL*** strands with no prevention on self-collision. Figure 2 shows 8 strands of $\boldsymbol{G_1}$ with same length generated by random selections of rewrite rules. Colliding blocks are shown in red in the images.

G_1: **X**
X → **X**h | **X**a | **X**d | **X**s | **X**t | **X**y | e

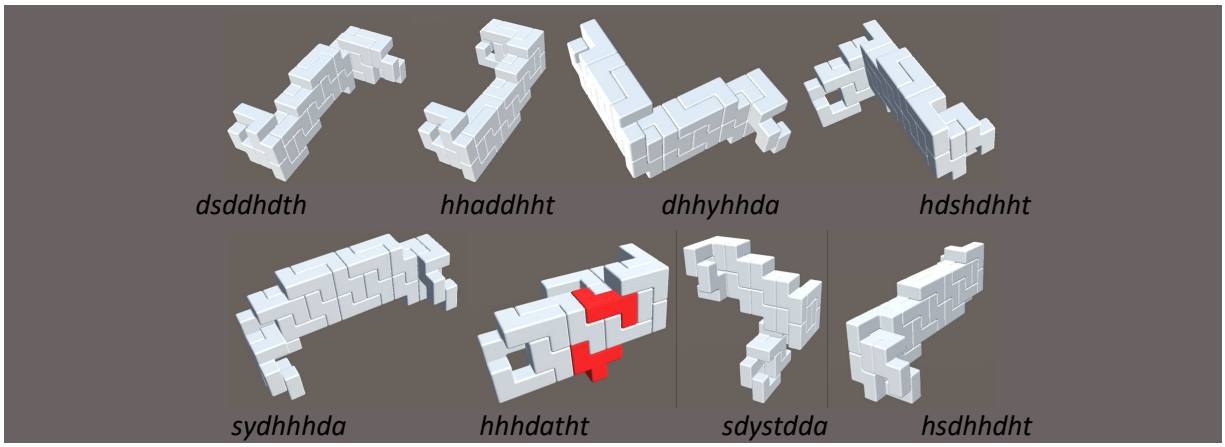


Figure 2: *8 strands from* G_1*, with red blocks showing self-collision*

The grammar G_2 defines looping strands that make elongated cubes of variable lengths. The language consists of palindromes with *aa* separating zero or even numbers of consecutive *h*'s on both sides and ends with an *a*. The simplest form in the language is a cube, denoted as the string *aaaa*.

G_2: **S**
S → a**B**a
B → hh**B**hh | **aa**

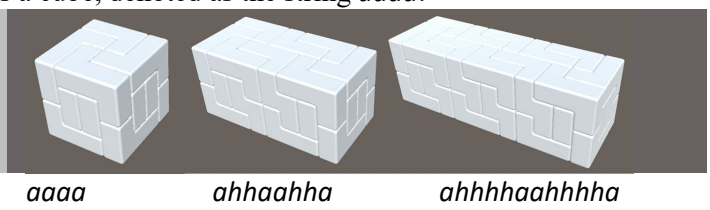


Figure 3: *3 strands of* G_2 *with various lengths*

The grammar G_3 extends G_2 by allowing the looping strands to make turns. Figure 4 shows the derivations of the strand *ahahahhaahahahha*.

G_3: **P**
P → a**L**a
L → h**S**h | aa
S → h**L**h | a**S**hah | hah**S**a

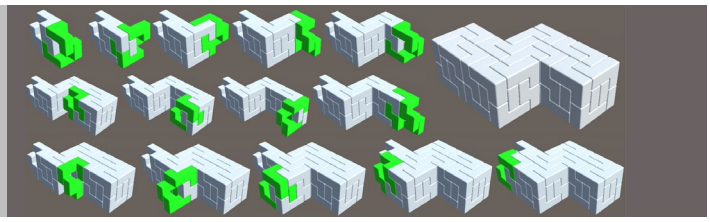


Figure 4: *The derivations of the strand* ahahahhaahahahha *from* G_3

The grammar G_4 generates **SL** strands with folded closed strands that might be regarded as "pseudo-branching", as shown in Figure 5.

G_4: **T**
T → a**L**a
L → h**S**h | aa
S → h**L**h | a**S**hah | hah**S**a | a**S**a**S**hh | hh**S**a**S**a | a**S**a**S**a**S**a

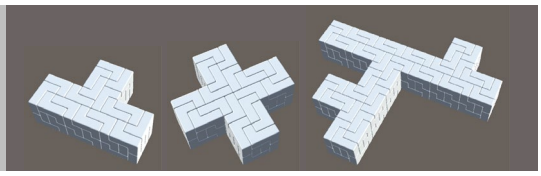


Figure 5: *Three strands from* G_5

A grammatical approach for form creation

Grammars can be used as high level abstraction of ***SL*** strand construction. In Figure 5, the grammar G_5 is used to create helical strands with various heights. The grammar G_6 is adapted from G_5 by inserting the second rule of G_2 with linkages to join the inserting strand created by the non-terminal ***B***. The grammar G_6 creates helical trunks with straight extrusions from four directions, as shown in the second row of Figure 5. The grammar G_7 uses rules in G_3 to allow changes of direction for branches. The grammar G_8 incorporates rules in G_4 to add extrusions to branches.

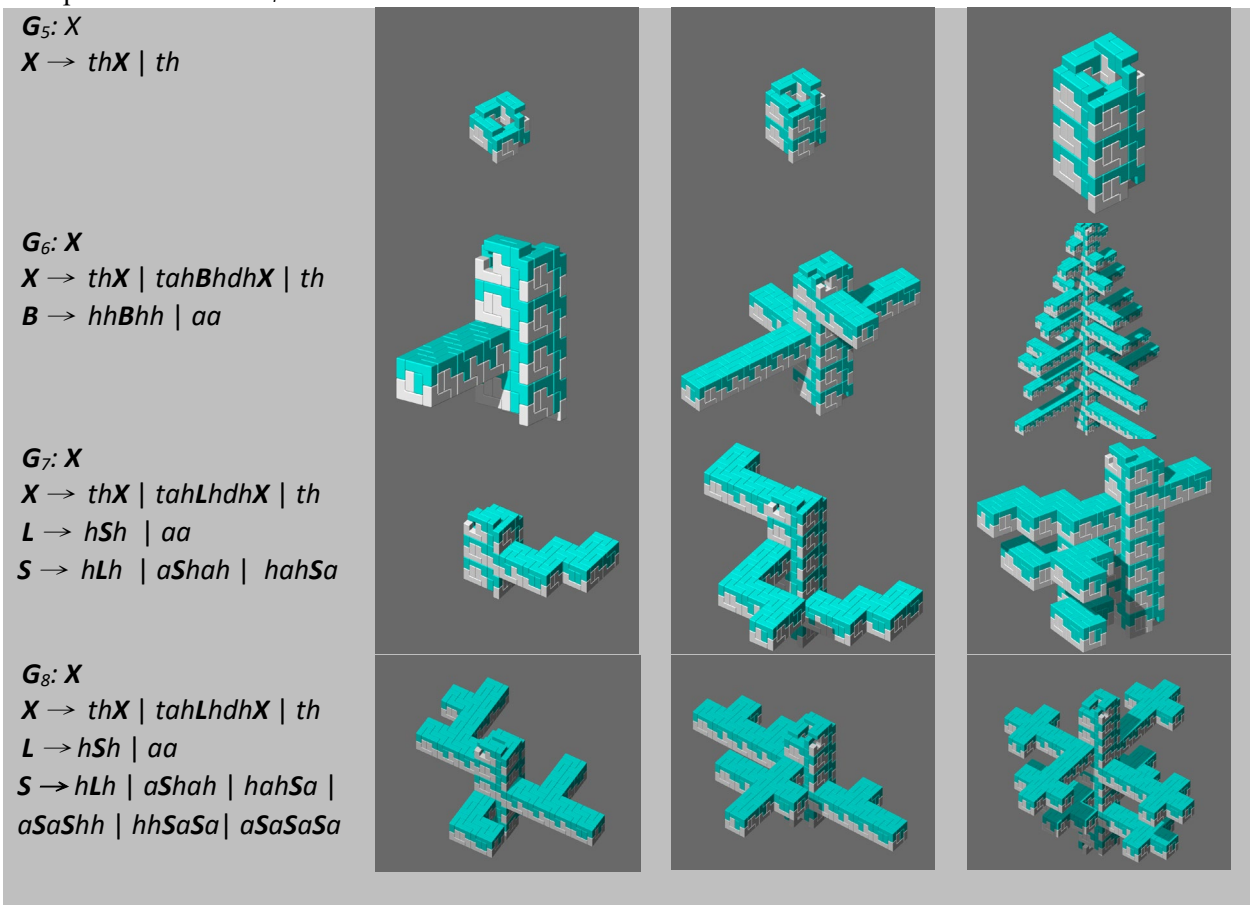


Figure 6: *Grammars for tree-like structures*

Guiding derivations with syntax-directed translations

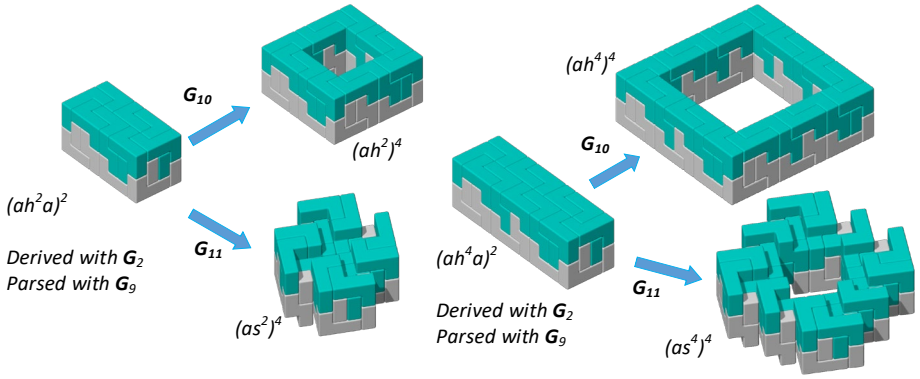
High level control on grammatical derivations can be helpful for artistic creation when some geometric patterns are preferred for form finding. The expressive power of context-free grammars is limited. For example, it is not possible to define non-trivial grammars that always generate rectangular forms with equal lengths for both pairs of opposite sides. Semantic checking with attribute grammars might be one way to bypass the limitation. In this study, syntax-directed translation is considered a means to go beyond the limitation. Figure 7 shows two translations that map strands derived by G_2 to strands defined by G_{10} and G_{11}. The grammar G_9 is the input grammar used to derive the parse tree that guides the derivations of G_{10} and G_{11} to create square forms. Strands generated by G_2 always have equal lengths on both sides of the folding structure. The grammar G_9 uses two non-terminals Y_1 and Y_2 to map the two sides of the input strand with identical derivations, which guide non-terminals Y_1 and Y_2 of G_{10} and G_{11} to generate four symmetrical sides for the square forms. The bottom row of Figure 7 shows a translation defined by inserting rules in G_9 to

replace some rules in G_6 for input grammar, and inserting rules in G_{11} to replace some corresponding rules in G_6 as the output grammar.

Input
G_9: X
$X \rightarrow aY_1aY_2$
$Y_{\{1,2\}} \rightarrow hhY_{\{1,2\}} \mid a$

Outputs
G_{10}: X
$X \rightarrow aY_1aY_2$
$Y_{\{1,2\}} \rightarrow hhY_{\{1,2\}}hh \mid a$

G_{11}: X
$X \rightarrow aY_1aY_2$
$Y_{\{1,2\}} \rightarrow ssY_{\{1,2\}}ss \mid a$



Input
G_{12}: X
$X \rightarrow thX \mid tahY_1aaY_2hdhX \mid th$
$Y_{\{1,2\}} \rightarrow hhY_{\{1,2\}} \mid e$

Output
G_{13}: X
$X \rightarrow thX \mid tat\ Y_1aY_2tadhX \mid th$
$Y_{\{1,2\}} \rightarrow ssY_{\{1,2\}}ss \mid a$

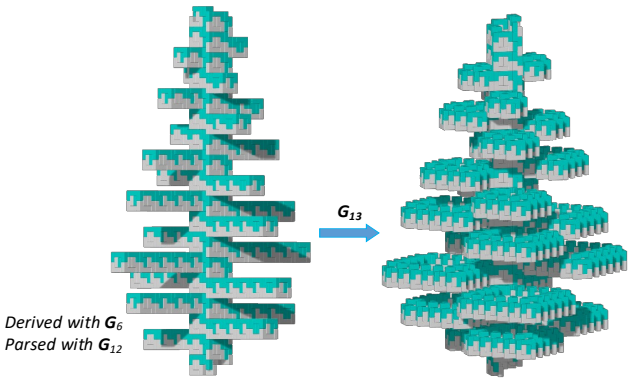


Figure 7: Examples of syntax-directed translations

Discussion

A straight forward mapping from strings to the construction of ***SL*** strands makes string grammars applicable for formalizing the representation, construction and analysis of ***SL*** strand constructions. With grammatical approaches, it might be possible to define high level operations and analysis that enable the design and assessment of very complicated constructions that are made of very simple and primitive elements such as ***SL*** blocks. It might not be possible to prevent spatial conflict in the rewrite rules of grammars, but it would be possible to incorporate collision detection in the generative process to prevent collisions. Future study may be directed towards the interactions between generative processes of grammars and the contextual environment of ***SL*** block construction.

References

[1] S. G. Shih, "On the Hierarchical Construction of SL Blocks – A Generative System That Builds Self-interlocking Structures." Sigrid Adriaenssens, F. Gramazio, M. Kohler, A. Menges, and M. Pauly Eds. *Advances in Architectural Geometry 2016*, Pages124-137, Hochschulverlag AG an der ETH Zürich, DOI 10.3218/3778-4, ISBN 978-3-7281-3778-4

[2] S. G. Shih, "The art and mathematics of self-interlocking SL blocks", *Bridges 2018 Conference Proceedings*, 107-114, 2018. http://archive.bridgesmathart.org/2018/bridges2018-107.pdf