Walking Around Trees: A 6-Letter 'DNA' for Baskets with Handles

James Mallos

Sculptor, Washington, DC, USA; jbmallos@gmail.com

Abstract

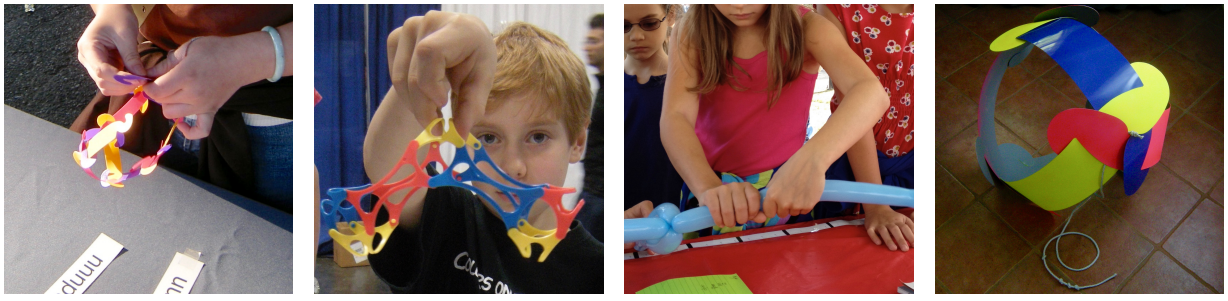
Fabric surfaces made by techniques like crochet and net making are typically worked in a linear order that meanders, without crossing itself, to ultimately visit and build the entire surface. For a closed basket whose surface is a topological sphere, it is known that the construction can be described by a codeword on a 4-letter alphabet via Mullin's encoding of plane graphs. Mullin's code exemplifies the formal language known as the Shuffled Dyck Language with 2 Types of Parenthesis (SDL_2.) Besides its 4-letter alphabet, SDL_2 has some other similarities to DNA: any word can be 'evolved' via sequence of local mutations (rewriting rules); and 'gene-splicing' two SDL_2 words by insertion or a concatenation, produces another SDL_2 word. But SDL_2 comes up short when we attempt to make a basket with handles. I show that extending the language with a third type of parenthesis works for orientable surfaces with handles provided an appropriate choice of *cutset* is made. (The cutset is the connected network of fronts—termed *cuts* in topology—where normal fabric construction encounters previously worked fabric.)

Introduction

As a sculptor, I have been interested in character sequences that code the construction of shapes. More accurately, the codes only specify fabric connections within the surface of the desired shape, but when fabric connections are made with uniform lengths, and an energy minimization is applied (for example, the natural tendency of the fabric in a basket to minimize its bending energy) a more or less definite shape results.

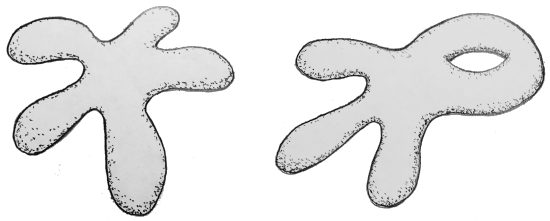
Some fun can be had making shapes this way. For example, I have done fair projects where participants learn to make a basket from a word (Figure 1a); or 'evolve' a new character sequence via rewriting rules, and then discover the shape it describes by snapping pieces together (Figure 1b,) or balloon twisting (Figure 1c); I have also made sculptures that include a description of their own shape using a knotted string code inspired by the khipu codes of the Incas.

The codes I have been using are able to describe the surfaces of branched 3-D objects like bushes, or bodies with arms and legs—which are topological spheres—but not surfaces with a topological handle (Figure 2).



(a) **(b)** **(c)** **(d)**

Figure 1: *Fun with building shapes from code: (a)"Make a Basket From a Word," (b)"Evolve a Basket," (c)"Evolve Your Own Balloon Animal," (d) "Knots Code Shapes."*



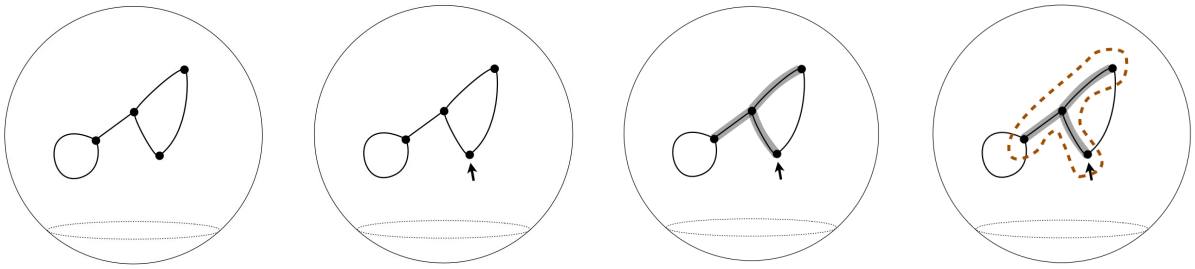
(a) **(b)**

Figure 2: *Basket topology: a closed basket whose surface has 0 handles (a), 1 handle (b).*

Background: Coding Baskets without Handles

Mullin [3], in 1967, described an encoding of graph drawings that has proved useful for coding the construction fabric surfaces. His interest was counting graphs drawn on the surface of the sphere. Informally, a graph drawing [4], on a surface is a connected drawing consisting of dots (0-dimensional vertices) connected by non-crossing curves (1-dimensional edges,) such that the regions left unmarked (2-dimensional faces,) are simply connected. All surfaces to be considered here will be orientable, and, in fact, endowed with an orientation that gives words like 'left' and 'counterclockwise' unambiguous meaning. All surfaces are initially closed, i.e., boundary-less, though artificial boundaries or *cuts* may be later introduced to render the surface into a topological polygon. (The baskets described by these surfaces are completely closed sculptural forms, lacking any opening to serve the usual functions of baskets.)

Mullin's project was counting tree-rooted plane graphs (Figure 3.) The term 'plane graph' is synonymous with 'graph drawing on the sphere'. A tree is a connected graph without cycles. A spanning tree of a graph, is a tree that is a subset of the graph and contains all of the vertices of the graph. One *roots* a graph drawing by distinguishing one of its corners—i.e., a *root face* together with a *root vertex*—by drawing a small arrow in the root face pointing toward the root vertex. By convention, the corner's edge that is counterclockwise from the rooting arrow is the *root edge*. By *tree-rooted* we mean that a graph drawing has been endowed with both a distinguished corner and a distinguished spanning tree. (Tree rooting gives the encoding algorithm a place to start, a direction to go in, and, as we will see, a complete ordering of the vertices and edges.)



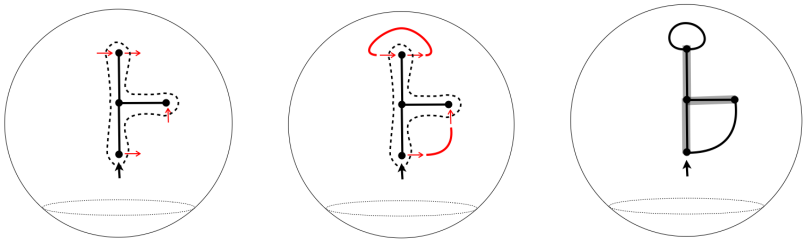
(a) **(b)** **(c)** **(d)**

Figure 3: *Tree-rooted plane graphs: (a) A plane graph, (b) a rooted plane graph, (c) a tree-rooted plane graph, (d) Mullin's encoding of a tree-rooted plane graph: starting from the root arrow and proceeding counterclockwise around the dashed contour, the encoding is '*([[])[()]]*'.*

Mullin's encoding algorithm (Figure 3d) begins at the root vertex and proceeds counterclockwise around a closed contour that wraps closely around the spanning tree. Walking around the contour, starting from where the rooting arrow intersects it, we will cross, and then re-cross, non-tree edges; and go along, and then

go back along, tree edges. These encounters are recorded in a character string using *two different styles* of parentheses: '(' records a crossing; ')' records a re-crossing; '[' records going out along an edge; ']' records going back along an edge.

The encoding tour in Figure 3d produces a *shuffled parenthesis word:* '([])[()]]'. Note that parentheses of each style, are, in relation to their own kind, properly nested, but the two different styles interleave freely. In this sense two distinct systems of parentheses have been shuffled.



(a) **(b)** **(c)**

Figure 4: *Mullin decoding: (a) Begin at what will be the root vertex, work in a counterclockwise circuit. Square parentheses direct drawing tree edges/moving along tree edges already drawn; round parentheses direct decorating vertices with outward/inward arrows. (b) Complementary arrows pair up to form the non-tree edges. (c) The recovered tree-rooted plane graph.*

The encoded tree-rooted plane graph can be recovered from its Mullin code through a sketching technique. Moving counterclockwise around the tree we are in the process of sketching (dashed trajectory in Figure 4a,) starting at a location that will become the root vertex: '[' says, "draw an edge;" ']' says, "move along the back side of an edge already drawn;" '(' says, "decorate the current vertex with an outward arrow;" ')' says, "decorate the current vertex with an inward arrow." On a second lap around the tree, complementary arrows are connected to recover the non-tree edges (red lines in Figure 4b.) The same rule used in algebra to pair up parentheses, *last opened, first closed*, is used to pair up outward and inward arrows. (As we see below, Mullin's sketching technique is not used in basket making, rather each of the four characters is interpreted as a craft action. The weaving, or other fabric technique, itself becomes the decoding.)

The Shuffled Dyck Language with Two Types of Parentheses

As a formal language, Mullin's encoding is SDL_2, the shuffled Dyck language with 2 types of parentheses. (The basic Dyck language, DL, has only one type of parenthesis, and so, necessarily, is un-shuffled.)

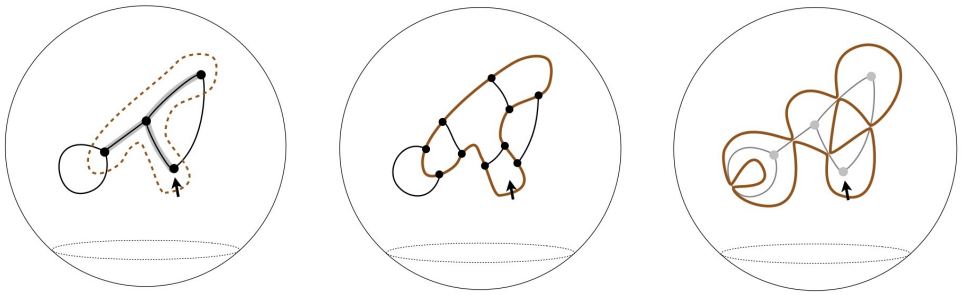
Starting from the *empty word* (i.e., a blank space) two rewriting rules or 'local mutations' suffice to generate all words in SDL_2:

> **Insert:** insert a matched pair, '()' or '[]', anywhere.
> **Shuffle:** shuffle adjacent parenthese of unlike type past each other, e.g., **[]()→[(])**.

These two rules combined allow us to build an SDL_2 word inside or beside another SDL_2 word, so SDL_2 words have a rather DNA-like property in that they can be freely concatenated or spliced into one another.

Using a Hamiltonian or Eulerian Working Order to Make Fabric

Walking around a tree may not seem like making fabric, but a Mullin code describes not only a tree-rooted plane graph (Figure 5a,) but also a *3-regular plane graph toured by a Hamiltonian circuit* (Figure 5b,) and a *4-regular plane graph toured by a non-crossing Eulerian circuit* (Figure 5c.) These two circuit interpretations serve as mathematical abstractions of the working order of a number of fabric techniques. For



(a) **(b)** **(c)**

Figure 5: *Mullin's encoding describes: (a) tree-rooted plane graph, (b) 3-regular plane graph with a rooted Hamiltonian circuit, (c) 4-regular plane graph with a rooted, non-crossing Eulerian circuit.*

example, unit weaving [1] and crochet suit the Hamiltonian interpretation of the Mullin code, net making and the scaffold-strand DNA technique [2] (now being used in nano-assembly) suit the non-crossing Eulerian interpretation.

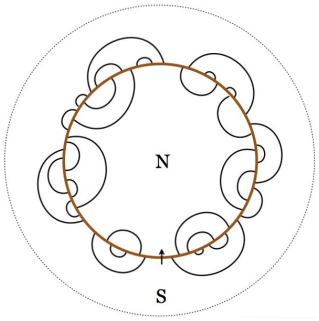


Figure 6: *Straightened out into a circle, the Hamiltonian circuit can be identified with the Equator—the square/round parenthesis pairs then represent non-crossing chords in the N/S hemispheres.*

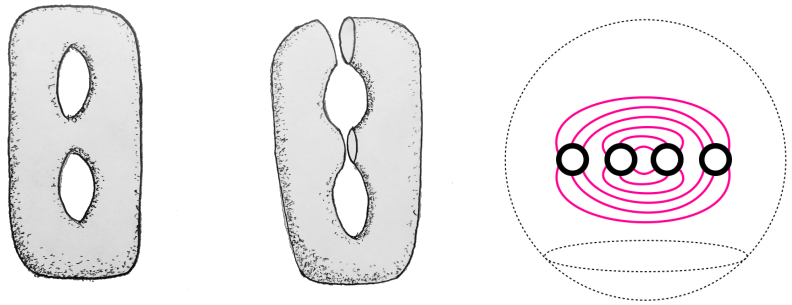
The Hamiltonian interpretation (Figure 5b) derives from the tree-rooted plane graph (Figure 5a) in the following way: in the interior of the coding contour (the brown loop,) construct a perpendicular across each tree edge; afterwards, erase everything else inside the interior of the contour. The two types of parenthesis are now seen to distinguish (see Figure 6) between chords on the Northern or Southern hemisphere of an Equatorial great circle.

The Eulerian interpretation derives from the Hamiltonian interpretation in the following way: imagine the chords in the Hamiltonian interpretation (Figure 5b) are elastic bands that shrink and pull their ends to their centers. We are left (see Figure 5c) with a non-crossing Eulerian circuit in a plane graph whose vertices are the centers of the edges of the original graph.

Coding Baskets with Handles

We aim to extend Mullin's code for topologically spherical baskets to baskets whose surfaces have an arbitrary number of topological handles. A surface with n handles is topologically equivalent to a 'tower' (Figure 7a) with n holes. Splitting the tower from the top to just shy of the bottom (Figure 7b) converts the surface into a sphere with $2n$ circular boundaries. Significantly, the identification of corresponding points on the

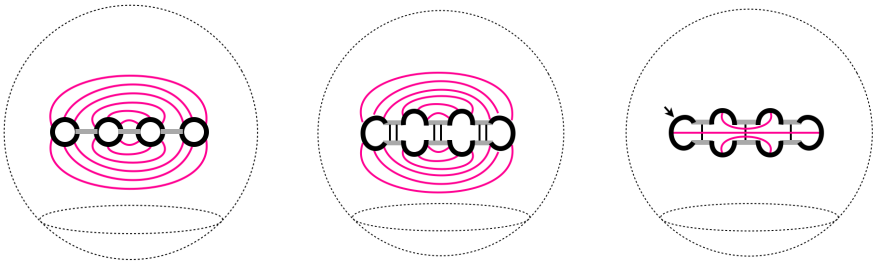
boundaries can be indicated by a set non-crossing 'field lines' drawn on the sphere (pink lines in Figure 7c.) We can now unite the boundaries into one unified boundary (a.k.a, polygonal schema) by making $2n - 1$ straight cuts which do not cross the field lines (Figure 8a.) The identification of corresponding points on the cuts can be indicated by non-crossing lines (narrow black lines in Figure 8b) drawn in the interior of the unified boundary.



(a) **(b)** **(c)**

Figure 7: *Converting an n-handle surface to a sphere with a structured arrangement of 2n boundaries: (a) an n-hole 'tower', (b) an n-hole tower split "from the top to just shy of the bottom,", (c) resulting sphere with 2n boundaries, curved lines indicate the identification of corresponding points.*

Taking up the Hamiltonian circuit interpretation of the Mullin code (Figure 5b,) the Hamiltonian circuit will not be allowed to cross the unified boundary (polygonal schema)—only non-Hamiltonian edges will be allowed to cross the boundary. In fact, our preference for counterclockwise travel around the Hamiltonian circuit means that only edges on the right side of the circuit can possibly cross the boundary. We will assume that such right-side, non-Hamiltonian edges are coded by pairs of round parentheses, '()'. We may therefore view the short black lines in Figure 8b as non-Hamitonian edges stretched across the opening boundary, and know that these are coded by pairs of round parentheses.



(a) **(b)** **(c)**

Figure 8: *Competing sets of non-crossing lines: (a) The 2n boundaries of Figure 7c unified by $2n - 1$ cuts, (b) unified boundary opened up (short black lines connect identified points on the cuts,) (c) simplified re-drawing with the 'field lines' inverted about the unified boundary.*

In Figure 8b, we can imagine inverting the pink field lines about the boundary, resulting in an equivalent set of non-crossing arcs drawn in the interior of the boundary. That imagined rearrangement makes two things clear: first, that we could code these connections with the same round parentheses used for the edges joining the cuts (narrow black lines in the Figure) and, second, if we did try to code both sets of connections with the same parenthesis style, the fact that the two sets of connections mutually cross would result in wrong

connections when we try to apply the *last opened/first closed* parenthesis rule. We need to innovate a special sort of round parenthesis, e.g., '{}', for the field lines. Having done so, we can deal with any number of handles as long as the unified boundary has been structured in this special way.

Figure 8c reduces the multiplicity of connections to a minimum so everything can be drawn in the interior of the boundary. A Hamiltonian circuit that is travelled counterclockwise will sample the unified boundary in what appears to be a clockwise order in this view. Starting at the arrow in Figure 8c, and proceeding clockwise around the boundary the indicated connections are: '({(}({)}{)})}'. Using unit weaving [1] of commercial *Flexeez* construction pieces, I have made the basket coded by that word (Figure 9). The surface is difficult to visualize in this rudimentary, skeletal form, but it is easy to verify that it is indeed orientable, has 12 vertices, 18 edges, and 4 faces. The Euler characteristic, $V - E + F = -2$, verifies that is indeed a double torus.

Figure 9: *A unit-woven double torus with the code word '({(}({)}{)})}'. Hamiltonian circuit in green, field lines pink, round-parenthesis edges black.*

Summary and Conclusions

I have shown that Mullin's 4-character encoding of tree-rooted plane graphs—which corresponds to mathematical abstractions of fabric construction—can be extended to orientable surfaces with handles provided we are free to make an advantageous choice of the polygonal schema.

My hope is that this extended code might have some of the 'DNA' like properties of SDL_2 for experiments in 'genetic engineering' of baskets with handles.

References

[1] J. Mallos. "Evolve Your Own Basket." *Bridges Conference Proceedings*, Towson University, Maryland, USA, July 25–29, 2012, pp. 575–580. http://archive.bridgesmathart.org/2012/bridges2012-575.html.

[2] J. Mallos. "DNA-Inspired Basketmaking: Scaffold-Strand Construction of Wireframe Sculptures." *Bridges Conference Proceedings*, Waterloo, Canada, July 27–31, 2017, pp. 57–62. http://archive.bridgesmathart.org/2017/bridges2017-57.pdf.

[3] R. C. Mullin. "On the Enumeration of Tree-Rooted Maps." *Canadian J. Math.*, vol. 19, no. 1, 1967, pp. 174–183.

[4] G. Schaeffer. "Planar maps." in *Handbook of Enumerative Combinatorics*, M. Bona, editor, CRC, 2015.