Flex Theory

Scott Sherman

Flexagons are interesting dynamic objects folded from strips of paper. You can decorate them with kaleidoscopic patterns or treat them as a puzzle. They were discovered in 1939 by Arthur H. Stone but popularized by Martin Gardner, whose very first Scientific American article was about the hexaflexagon[1]. A later Gardner column described its cousin, the square tetraflexagon[2].

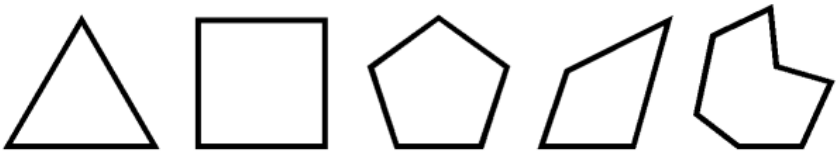
Since then, many generalizations of flexagons have been explored. Flexagons have been made out of a variety of triangles, squares, trapezoids, pentagons, hexagons, and other polygons.[3] Different numbers of triangles arrayed around the center create the triangle pentaflexagon, octaflexagon, dodecaflexagon, etc.[4] And new patterns of folding, or "flexes," have been discovered for manipulating these flexagons.[5]

This paper will specifically discuss flexagons made from various triangles. It will describe different flexes, most of which have been discovered (or at least named) by the author, as well as a theory that can be used to create and explore these flexagons. Specifically, it will define *flex notation* and *pat notation*. Flex notation can be used to describe a series of flexes. Pat notation can be used to describe how the internal structure of the flexagon changes as flexes are applied. Pat notation can also be used for creating new structure that allows for a specific series of flexes.

Definitions

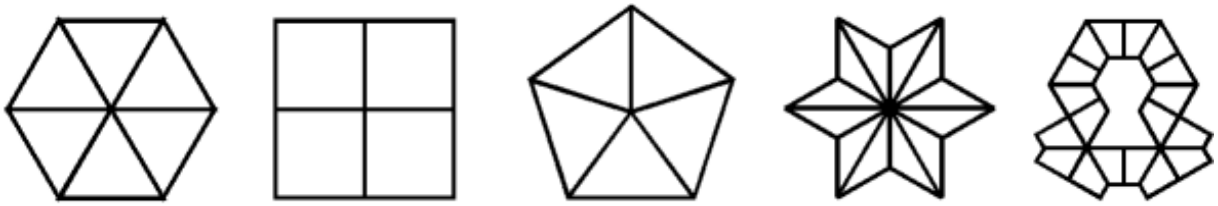
leaf: *A single polygon*

Here are some examples of leaves:



***f-linkage**: A chain of leaves where every leaf is connected along exactly two edges to mirror images of itself*

Note that an f-linkage does not need to be planar or symmetrical, and may contain twists. Here are some examples of f-linkages:



An unfolded flexagon is an f-linkage. The shape of a folded flexagon is also an f-linkage.

***pat**: A portion of an f-linkage where the leaves are folded along edges such that the mirrored edges of the leaves all align*

A pat is a stack of leaves. Typically, one edge of the unfolded f-linkage is disconnected so the pats can be folded. After folding, the first and last edges are connected together to restore the f-linkage.

Two important characteristics of a flexagon are that of all the leaves are identical and every leaf is connected to exactly two other leaves mirrored across edges.

To describe a flexagon made from a particular polygon, I use the polygon name as a prefix, e.g., triangle flexagon, square flexagon, or pentagon flexagon. I use a Greek prefix to indicate the number of polygons per face, e.g., tetraflexagon for four polygons per face and hexaflexagon for six polygons per face. A second Greek prefix can be used to indicate the number of sides you can pinch flex to, e.g., pentahexaflexagon for a hexaflexagon with five sides. But, as we'll see, this pinch-flex-centric concept of "side" doesn't translate to other flexes.

So now that you have a basic understanding of a flexagon, what can you do with it?

***flex**: A series of modifications to a flexagon that takes it from one valid state to another, where the modifications consist of folding together, unfolding, and sliding pats*

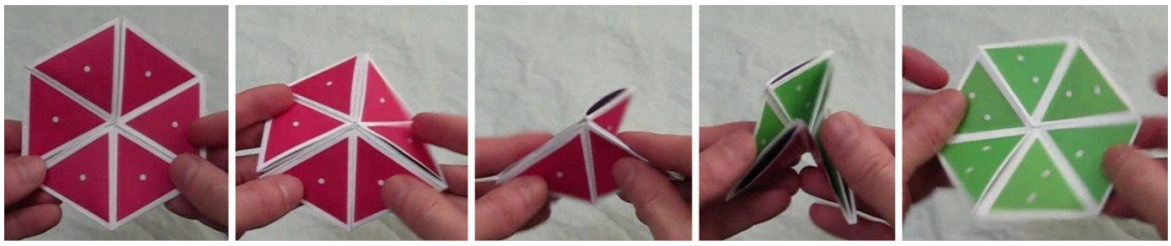
A flexagon is in a valid state when it has been created through leaf splitting, described later.

Flexes

The following sections show some of the possible flexes on triangle flexagons.

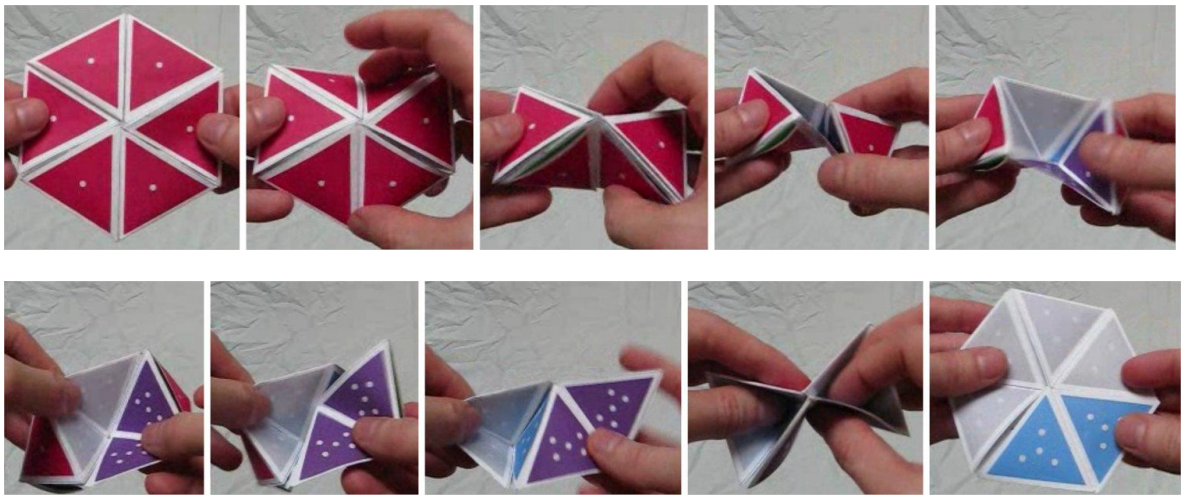
Pinch flex

With the standard *pinch flex* on a hexaflexagon, you start by folding together adjacent pairs of pats, turning the original six pats into three pats. You then unfold each of those pats along different edges, restoring it to six pats, but in a different configuration.



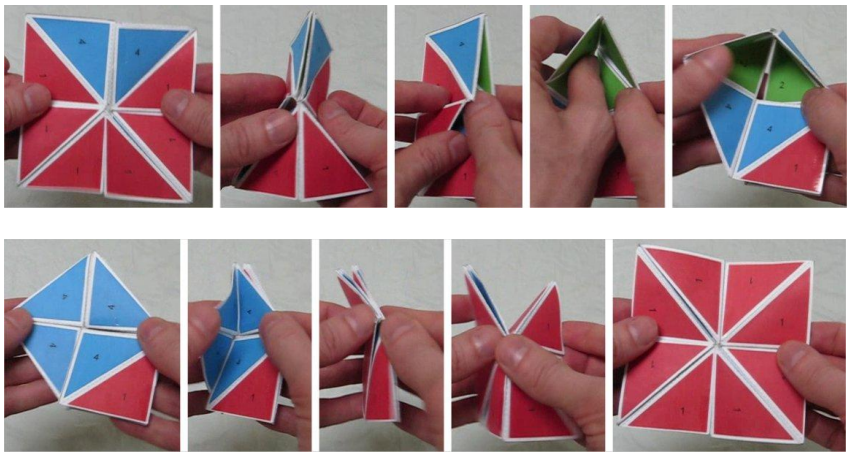
V-flex

The *v-flex* was discovered by Bruce McLean[6]. To perform the v-flex, fold together opposite vertices, open up the flexagon in the center, then slide the pats before opening it up again.



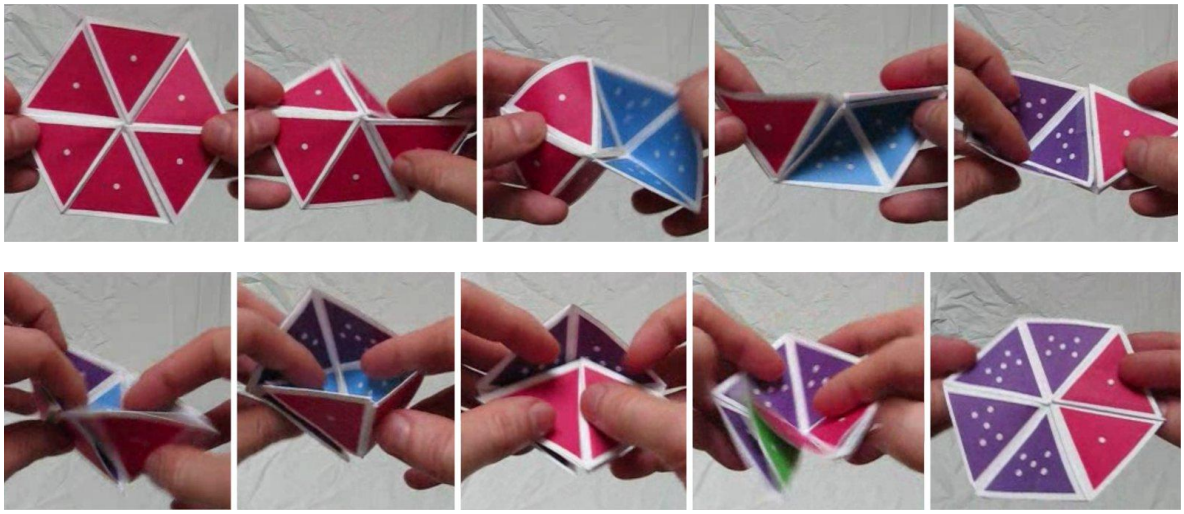
Tuck flex

The *tuck flex* was dismissed by Conrad and Hartline as a "distortion" and "a relatively well known but much dreaded aspect of the flexagon."[7] This author finds it an interesting flex that adds a lot to the exploration of flexagons. It is a particularly elegant flex on flexagons made of right triangles, where no "distortion" is required. To perform it, you fold the flexagon in half, "tuck" a pair of leaves in the fold, then open it up again.



Pyramid shuffle flex

The *pyramid shuffle flex*, discovered by the author, works on one or two pairs of adjacent pats at a time without impacting the entire flexagon. You alternately fold pats together and unfold them from a different set of hinges.



Flex notation

Flex notation describes a series of flexes and where you perform each flex. Each flex has a notion of a *current vertex* and *current side* that the flex is performed relative to. The current vertex and side can be changed as follows:

^	Change the current side. Flip the flexagon over while keeping the same current vertex.
<	Change the current vertex. Step one vertex counterclockwise (left if vertex is at the top).
>	Change the current vertex. Step one vertex clockwise (right if vertex is at the top).

Here are the symbols for some of the possible flexes:

I	Identity flex
P	Pinch flex
T	Tuck flex
S	Pyramid shuffle flex
V	V-flex
F	Flip flex
St	Silver tetra flex
Lt	Slot-tuck flex
Ltb	Slot-tuck-bottom flex
Tk	Ticket flex

This paper does not describe how to physically perform all of these flexes beyond the photos above. The author's website, http://loki3.com/flex/flex/, has more information about each of these flexes, including videos and instructions for making flexagons that support the flexes.

Other notation:

(...) x n	Repeat a series of flexes n times, e.g., (P<) x 2 means repeat the sequence P< twice
X'	Perform the inverse of flex X, i.e., the exact opposite operation

Now you can start to make interesting statements such as P' is equal to ^P^ and I is equal to AA', where A represents any flex or sequence of flexes.

One thing important to note here, which will make more sense after reading the next section, is that you can't always perform a particular flex at a given vertex. For example, P transforms a flexagon in the same way the flex sequence TV does. However, you can't always carry out P in the same place you can do TV, and vice versa. This leads to the following notation:

A = B	The flex sequences A and B always transform the flexagon in the same way
A ≈ B	A and B transform the flexagon in the same way as long as the pat structure allows it

Thus you would write P' = ^P^ because the equality always holds, but P ≈ TV because being able to perform the flexes on one side of the ≈ doesn't guarantee that the flexes on the other side can be performed.

Inverse equalities:

^'	=	^	
<'	=	>	
>'	=	<	
P'	=	^P^	
T'	≈	^T^	note the use of ≈ rather than =
S'	=	^>S^>	unusual because you need to shift the current vertex
V'	=	^V^	
F'	=	^F^	
St'	=	^St^	
Lt'	=	^Lt^	

General identities:

I = AA'		
(AB)' = B'A'		i.e., you can undo two flexes by undoing the second flex then undoing the first
I = AB	=>	A = B'
I = ABC	=>	A = C'B' C = B'A'

Selected flex formulas, where *n* is the number of pats in the flexagon:

I ≈ SP<T>P' S ≈ P<^T<P^ T ≈ <P^SP^>

I ≈ V>>>TP'>>> P ≈ >>>V>>>T V ≈ >>>PT'>>>

I ≈ F^SSt^ St ≈ F^S^ F ≈ St>S< S ≈ FSt'

St ≈ >T'<<T'>

PP ≈ (T>>) x n/2

P^>P^>P ≈ (S<<) x n/2

I ≈ (S>T'>^T^>>) x 2

Pat notation

Recall that a *pat* is a stack of connected leaves in a flexagon. A hexaflexagon has six pats, for example.

When describing a pat in *pat notation*, the leaves are listed from top to bottom, with parentheses around them indicating grouping. An individual leaf is given a single label rather than each side having a separate label. Leaves are grouped into pairs based on the rules of leaf splitting, explained in the *Leaf splitting* section below. Thus a single leaf pat is (a). After leaf splitting, it could be described as (a,b). This notation represents two leaves connected to each other by a single edge, folded together such that leaf *a* is on top and leaf *b* is below it.

If the top leaf is split next, the result is ((a,b)c). This notation represents the two leaves *a* and *b* hinged together with another hinge connecting that top sub-pat to leaf *c* on the bottom. If the bottom leaf is split, the result is (a(b,c)). The letters in these examples can be considered arbitrary labels for the leaves. The order in which the leaves appear in the physical pat matches the order listed in pat notation.

There are five different pats with four leaves: (((a,b)c)d), (a(b(c,d))), ((a,b)(c,d)), ((a(b,c))d) and (a((b,c)d)). It can be shown that the number of possible pat structures for n leaves is the Catalan number C_n[8].

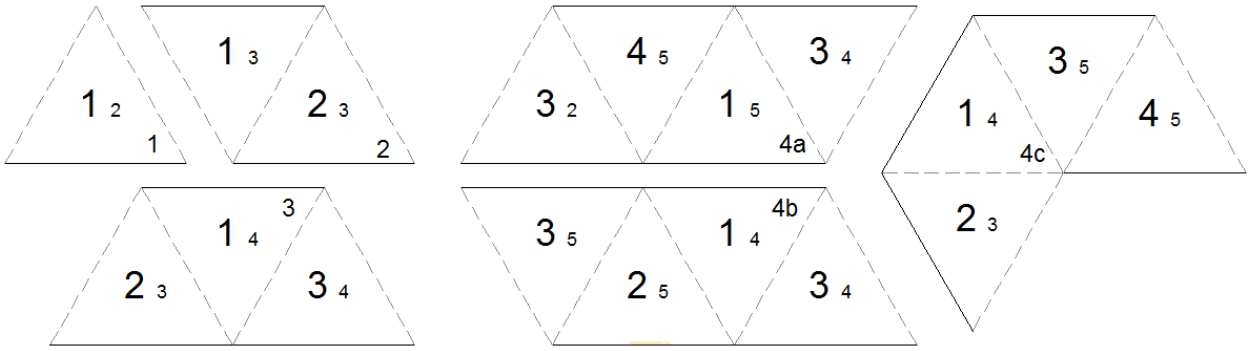
In this paper, I use lower case letters to represent a leaf or sub-pat. Numbers are used in specific flexagons to represent a single leaf. The following symbols are used to indicate specific pat structures:

1	(a)	4a	(((c,^b)^d)a)
2	(^b,a)	^4a	(^d(a(c,^b)))
3	((^b,c)a)	4b	((c,^d)(a,^b))
^3	(c(a,^b))	4c	((^b(^d,c))a)
		^4c	(^d((^b,a)c))

In the above definitions, the leaves are assigned letters in the order they appear in the unfolded template, with *a* coming first, *b* second, and so on. *^b* indicates that leaf b is upside down. While this ordering is not a requirement for pat notation, it makes it more apparent how the flexagon is folded.

Shown below are templates you can use for creating each of these pats. When the table above uses a caret (^), the given pat is turned over (e.g., ^3 indicates that 3 should be turned over). The small

number on each leaf goes on the back. When folding the pat, fold like numbers against adjacent like numbers, starting with the highest numbers first. Cut along solid lines. Dashed lines indicate a hinge, either connecting to another leaf in the same pat or an adjacent pat.



A flexagon is described by listing its pats in clockwise order. For example, the trihexaflexagon can be described as (^2,1) (^3) (5,^4) (6) (^8,7) (^9). The *current vertex* is the one between the first and last pats in the description. In this example, the current vertex is between (^9) and (^2,1).

Now we can start defining manipulations on a flexagon. Changing the current vertex is done by simply rotating the positions of the pats. For example, we can apply the > and < operators (shift the current vertex one to the right or one to the left, respectively) to the trihexaflexagon like so:

(1,2) (3) (4,5) (6) (7,8) (9)	>	(3) (4,5) (6) (7,8) (9) (1,2)
(1,2) (3) (4,5) (6) (7,8) (9)	<	(9) (1,2) (3) (4,5) (6) (7,8)

^ (turning over the flexagon) is defined by reversing the order of the pats and the structure within each pat. For instance, turning over the pat (1(2(3,4))) gives you (((^4,^3)^2)^1), where ^1 indicates that leaf 1 is upside down. This is especially important to note when the pat has substructure. The following is an example of turning over a sample tetraflexagon:

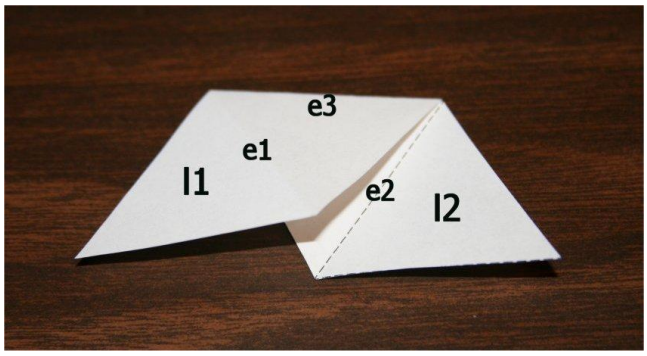
(1) (2,3) (4(5,6)) ((7,8)(9,10))	^	((^10,^9)(^8,^7)) ((^6,^5)^4) (^3,^2) (^1)

Sequences of operations can easily be applied one after another.

Leaf splitting

Before continuing with more pat manipulations, it is useful to have a description of *leaf splitting*, mentioned above. This technique will allow us to go from a flexagon described in pat notation to a physical model.

Consider a leaf l_0 in a pat. It's connected to leaf l_1 across edge e_1 and to leaf l_2 across edge e_2, leaving its third edge e_3 unconnected. If you think of the leaf as having a small amount of thickness, you can slice it in two, leaving the two new leaves connected across e_3. One of the new leaves is connected across e_1 to l_1 while the other is connected across e_2 to l_2.



There are two ways to perform this leaf splitting, with either the top leaf connected across e_1 and the lower leaf across e_2 or vice versa. The choice you make for the first leaf splitting operation determines the handedness of the flexagon. Every leaf splitting operation thereafter must use the same handedness; otherwise you no longer have a valid flexagon.

A consistent handedness is guaranteed when using the pat templates from the previous section. Since those templates are sufficient for of all the flexes and techniques in this paper, a general description of handedness won't be included. The interested reader can refer to the Maps and Plans section of the Conrad and Hartline paper[9] for a different approach to leaf splitting.

Now that we have a concept of handedness and leaf splitting, we can provide a definition of a flexagon.

> ***flexagon***: *An f-linkage of pats where every pat and sub-pat has been created through leaf splitting and has the same handedness.*

Flex definitions

> ***minimal flexagon***: *The simplest pat structure that supports a given flex or set of flexes.*

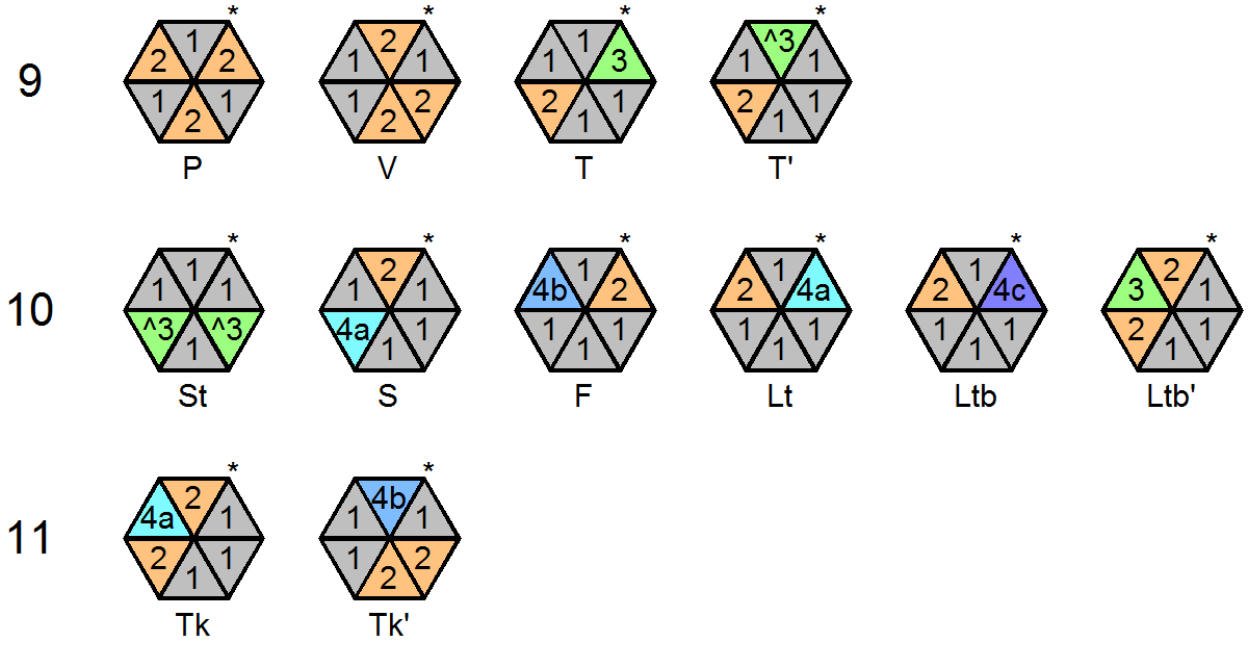
We are now ready to define various flexes. All of these flexes are defined in terms of the *minimal flexagon* for the given flex. We will start with definitions that apply to the triangle hexaflexagon before generalizing to triangle flexagons with different numbers of pats.

(^b,a) (^c) (e,^d) (f) (^h,g) (^i)	P ->	(b) (^d,c) (^e) (g,^f) (h) (a,i)
((^b,c)a) (d) (e) (^g,f) (^h) (^i)	T ->	(c) (d) (e) (^g,f) (^h) (b(^i,^a))
(^b,a) (^c) (^d) (^e) (((^h,g)i)^f) (j)	S ->	(^b(i(a,^j))) (^c) (^d) (^e) (g,^f) (h)
(a) (^c,b) (e,^d) (f) (g) (^i,h)	V ->	(b,^a) (c) (d) (^f,e) (h,^g) (i)
(^b,a) (^c) (^d) (^e) ((^h,i)(^f,g)) (j)	F ->	(i) ((b,^c)(j,^a)) (^d) (^e) (^f) (h,^g)
(c(a,^b)) (d) (e) (f) (i(g,^h)) (j)	St ->	(a) ((^c,d)b) (e) (f) (g) ((^i,j)h)
(((c,^b)^d)a) (^e) (^f) (^g) (i,^h) (j)	Lt ->	(j) (^b,a) (^c) (^d) (^e) (i(^f(^h,g)))
((^b(^d,c))a) (^e) (^f) (^g) (i,^h) (j)	Ltb ->	(^a) (^b) (^c) (e,^d) ((^g,h)f) (^j,i)
(a) (b) (c) (^e,d) (((^h,g)i)^f) (^k,j)	Tk ->	(^h) (f,^g) (^d,e) (^c) (^b) ((j,^i)(^a,^k))

Note that the silver tetra and flip flexes don't actually work on an equilateral triangle hexaflexagon (at least not without lots of protest). They do work nicely on a right triangle hexaflexagon, isosceles triangle

octaflexagon, and many others; thus, they are included here for completeness. Some flexes require mild bending of leaves on an equilateral triangle flexagon, but not on many other triangle flexagons.

The following table shows another way to understand the required pat structure for these flexes. The number at the beginning of each row represents the number of leaves in the minimal flexagon. The asterisk marks the current vertex. The symbols on each pat denote the pat's structure and are described in the *Pat notation* section above. Most flexes have the same structure before and after the flex. However, the flexagon has different structure after applying T, Ltb, and Tk; hence their inverses are listed as well.



When these transformations are applied to flexagons with deeper pat structure, the sub-pat structure is preserved. As an example, let's start with one configuration of a pentahexaflexagon.

(((3,^2)^4)1) (5) (((8,^7)^9)6) (10) (((13,^12)^14)11) (15)

Let's look again at the definition of the pyramid shuffle.

(^b,a) (^c) (^d) (^e) (((^h,g)i)^f) (j) S -> (^b(i(a,^j))) (^c) (^d) (^e) (g,^f) (h)

Comparing the pat structure of the pentaflexagon state above with the before state of the pyramid shuffle, we can see that they align in the following way:

a	1	^f	11
^b	((3,^2)^4)	g	^12
^c	5	^h	13
^d	(((8,^7)^9)6)	i	^14
^e	10	j	15

Thus applying the pyramid shuffle to the pentahexaflexagon state gives us the following result:

(((3,^2)^4)(^14(1,^15))) (5) (((8,^7)^9)6) (10) (^12,11) (^13)

Now look at the definition of the pinch flex.

(^b,a) (^c) (e,^d) (f) (^h,g) (^i) P -> (b) (^d,c) (^e) (g,^f) (h) (a,i)

You can see that it is possible to apply a pinch flex to the previous result, noting that *a* corresponds to ((3,^2)^4), *b* to (^14(1,^15)), *c* to 5, *d* to ((8,^7)^9), and so on. This gives you the following state:

(4(2,^3)) (6,^5) (9(7,^8)) (11,^10) (12) ((^14(1,^15))13)

You could continue this way, rotating, flipping, and applying flexes, to explore every state of the pentahexaflexagon.

As a second example, we'll show that Tk, the ticket flex, is always equivalent to Ltb'T<<V^. First we look at what the ticket flex does:

		(1) (2) (3) (^5,4) (((^8,7)9)^6) (^11,10)
Tk	->	(^8) (6,^7) (^4,5) (^3) (^2) ((10,^9)(^1,^11))

Now apply Ltb'T<<V^ to the same starting pat structure:

		(1) (2) (3) (^5,4) (((^8,7)9)^6) (^11,10)
Ltb'	->	((2(4,^3))^1) (5) (6) (^8,7) (10,^9) (11)
T	->	(4,^3) (5) (6) (^8,7) (10,^9) (^2(11,1))
<<	->	(10,^9) (^2(11,1)) (4,^3) (5) (6) (^8,7)
V	->	((11,1)(9,^10)) (2) (3) (^5,4) (7,^6) (8)
^	->	(^8) (6,^7) (^4,5) (^3) (^2) ((10,^9)(^1,^11))

This final result matches the final state of the ticket flex. This proves that Tk = Ltb'T<<V^.

Flex generating sequences

If the required pat structure of a flex is satisfied by a flexagon, you can simply apply the flex to transform the state of the flexagon as demonstrated in the previous section. But if a flex is not supported, the flex's pat structure can be used as a recipe for modifying the flexagon to allow you to apply the flex.

For the simplest example, consider a hexaflexagon where every pat consists of a single leaf.

(1) (2) (3) (4) (5) (6)

Obviously, you can't use the pinch flex on it (or any other flex, for that matter). But you *can* use the definition of the pinch flex to create the minimal flexagon that supports the flex. We'll use the "after" state of the flex so that we've essentially applied the pinch flex.

P -> (2) (^4,3) (^5) (7,^6) (8) (1,9)

When you start from the degenerate flexagon with single leaf pats and apply flex definitions to create a new flexagon, this series of flexes is called the *generating sequence* for the flexagon. In this case, we created a flexagon from the generating sequence P.

Comparing the results of the pinch flex above with the required structure for a tuck flex, ((^b,c)a) (d) (a) (^g,f) (^h) (^i), we see that we need to add new leaves in the first pat in order to perform the flex. Creating the new leaves (then renumbering to match the order they appear in the unfolded template) gives us the following:

T -> ((^2,3)1) (^5,4) (^6) (8,^7) (9) (^11,10)

This flexagon has the generating sequence PT. If we were going to continue on, we would want to apply the tuck flex to this new structure.

Traditional flexagons use a generating sequence consisting entirely of pinch flexes combined with rotations or flips. Elegant theory has been developed to analyze these flexagons and traversals of all of the states accessible using only the pinch flex.

But using other flexes to generate flexagons opens up a lot of new areas to explore. As an example of what else we can do, we just created an 11-leaf hexaflexagon from the generating sequence PT. This flexagon has 13 different states and also supports the v-flex, pyramid shuffle, slot-tuck, and slot-tuck-bottom flexes. Seven of these states don't support the pinch flex.

General triangle flexagons

The above flex definitions were given for the hexaflexagon, but those flexes can all be generalized to flexagons with different numbers of triangles per side. Some operate on the entire flexagon, e.g., the pinch flex, while others operate on only a portion of the flexagon, e.g., the pyramid shuffle.

Here are the definitions of the pinch flex for the triangle tetraflexagon (four triangles per side) and triangle octaflexagon (eight triangles per side):

(^b,a) (^c) (e,^d) (f)	P ->	(b) (^d,c) (^e) (a,^f)
(^b,a) (^c) (e,^d) (f) (^h,g) (^i) (k,^j) (l)	P ->	(b) (^d,c) (^e) (g,^f) (h) (^j,i) (^k) (a,^l)

The following is the definition of the pyramid shuffle on a triangle pentaflexagon (five triangles per side):

(^b,a) (^c) (^d) (((^g,f)h)^e) (i) S -> (^b(h(a,^i))) (^c) (^d) (f,^e) (g)

The tuck flex is interesting in that the fundamental requirement is a pat with the structure ((1,2)3) and enough freedom elsewhere in the flexagon that this pat can be opened up. On a hexaflexagon, there is only one way to achieve this: by having a hinge on the opposite side of the current vertex. On an octaflexagon, there are multiple ways this could work:

((^b,c)a) (d) (e) (^g,f) (^h) (^i) (^j) (^k)	T ->	(c) (d) (e) (^g,f) (^h) (^i) (^j) (b(^k,^a))
((^b,c)a) (d) (e) (f) (^h,g) (^i) (^j) (^k)	T ->	(c) (d) (e) (f) (^h,g) (^i) (^j) (b(^k,^a))
((^b,c)a) (d) (e) (f) (g) (^i,h) (^j) (^k)	T ->	(c) (d) (e) (f) (g) (^i,h) (^j) (b(^k,^a))

Here are some general definitions for flexes on triangle flexagons, where n is the number of leaves:

(1,2) (3) … (i,i+1) (i+2) … (n-2,n-1) (n)	P ->	(^1) (5,^3) … (^i) (i+4,^i+2) … (^n-2) (2,^n)
((1,2)3) (4) … (i,i+1) … (n-1) (n)	T ->	(2) (4) … (i,i+1) … (n-1) (^1(n,^3))
(1,2) (3) … (i) … (((n-4,n-3)n-2)n-1) (n)	S ->	(1(n-2(2,^n))) (3) … (i) … (n-3,n-1) (^n-4)
(1,2) (3) … (i) … ((n-4,n-3)(n-2,n-1)) (n)	F ->	(n-3) ((^1,3)(n,^2)) … (i) … (n-2) (^n-4,^n-1)
(1(2,3)) (4) … (i) … (n-3(n-2,n-1)) (n)	St ->	(2) ((^1,4)^3) … (i) … (n-2) ((^n-3,n)^n-1)
(((1,2)3)4) … (i) … (n-4) (n-3) (n-2,n-1) (n)	Lt ->	(n) (2,4) (^1) (3) … (i) … (n-2(n-4(n-1,^n-3)))

Exploration

As an example of what we can do with this notation, we will look at all of the states of the three standard six-sided hexaflexagons (hexahexaflexagons) and the number of flexes supported by the states. I will refer to the flexagons by the flex generating sequences (P'>PP)x2, (P^>)x4, and Px4 where the first one corresponds to variation A at http://flexagon.net/, the second to variation B, and the third to variation C. Exploration of these flexagons using the flexes in this paper reveals that the first has a total of 3420 states, the second has 2358 states, and the third has only 513 states.

For each of these three hexahexaflexagons, the following table lists the number of states of the flexagon that support each flex (the "states" column) and the total number of times each flex can be performed (the "total" column). Inverse flexes are listed when the pat structure is different before and after the flex. It is interesting to note that every state of variation A supports the pinch flex while less than 8% of the states in variation C do. The relative frequency of occurrence of the flexes varies significantly between variations.

	(P'>PP)x2		(P^>)x4		Px4	
	states	**total**	**states**	**total**	**states**	**total**
P	3420	3426	438	456	39	48
T	1788	4644	1950	3984	231	708
T'	3024		2346		399	
Ltb	2910	4140	1376	1584	91	120
Ltb'	1788		1014		60	
Lt	2910	2070	800	504	148	96
S	2910	2070	1652	1056	352	216
V	1788	2322	2310	2328	117	132
St	774	540	780	540	399	264
Tk	558	576	844	1116	30	42
Tk'	276		924		30	
F	528	540	1204	852	242	156

For a second example, we look at what it takes to create a simple cycle of flexes that brings you back to your original state. We would like to be able to apply the same flex multiple times, possibly rotating the flexagon but not turning it over. In other words, we're looking to satisfy the equation I = (A(> x a)) x b for some flex A and numbers *a* and *b*. There are very few solutions. The following table shows some results, listing the flex sequence to repeat, the number of repetitions in the generating sequence, the number of repetitions in the full cycle, and the pat structure of the minimal flexagon. The final column uses shorthand notation for describing the pats.

flex	gen	cycle	flexagon	
P>	1	3	(1) (^3,2) (^4) (6,^5) (7) (^9,8)	1 2 1 2 1 2
V>	4	6	(^2,1) (^3) (5,^4) ((8,^9)(6,^7)) (11,^10) (12)	2 1 2 4b 2 1
V<	7	15	((3,^4)(1,^2)) (6,^5) (7) (^9,8) ((^12,13)(^10,11)) (^15,14)	4b 2 1 2 4b 2
F<	3	4	(1) ((4,^5)(2,^3)) ((^8,9)(^6,7)) ((12,^13)(10,^11)) (^14) (16,^15)	1 4b 4b 4b 1 2
F>>>	7	8	(1,2) ((3(4,5))((6,7)8)) (9,10) (11,12) (13,14) (15,16)	2 (^33) 2 2 2 2
Ltb<	2	3	(1) (2) (3) (^5,4) ((7,^8)^6) ((10(12,^11))^9)	1 1 1 2 3 4c
Ltb>>>	4	15	(1) (2,3) (4) (((5,6)(7,8))9) ((10,11)(12,13)) (14,15)	1 2 1 (4b1) 4b 2

Many of these sequences apply to other flexagons as well. The following table shows some examples for octaflexagons:

flex	gen	cycle	flexagon
P>	1	3	(1) (^3,2) (^4) (6,^5) (7) (^9,8) (^10) (12,^11)
F<	5	6	(1) ((4,^5)(2,^3)) ((^8,9)(^6,7)) ((12,^13)(10,^11)) ((^16,17)(^14,15)) ((20,^21)(18,^19)) (^22) (24,^23)
F>>>	9	10	(1,2) ((3(4,5))((6,7)8)) (9,10) (11,12) (13,14) (15,16) (17,18) (19,20)

Further exploration

It has been shown that every state of a pentahexaflexagon can be reached using P, T, T', S, and Ltb[10]. But there are also flexagons, such as ((^2,3)1) (4) (5) (6) (7) (8), that can't be fully explored using the flexes listed here (though a "forced tuck" could be used in this example). Under what conditions can every state of a flexagon be reached using the flexes in this paper? Are there additional flexes that open up new states on some triangle flexagons?

The Tuckerman traverse defines a general technique for visiting every available state using the pinch flex. Are there similar recipes for traverses using other flexes or sets of flexes?

If leaves are mirrored across points rather than edges, you get point flexagons[11]. Pat notation could easily be extended to cover them as well.

Flexagons can be made from polygons other than triangles. How does the notation in this paper need to change to include square flexagons, pentagon flexagons, etc.?

Pat notation offers a simple technique for automated exploration of flexagons using binary trees. What patterns are out there to discover?

References

[1] *Hexaflexagons and Other Mathematical Diversions* by Martin Gardner, 1959.
[2] *The Second Scientific American Book of Mathematical Puzzles and Diversions* by Martin Gardner, 1961.
[3] For example, see *Flexagons Inside Out* by Les Pook, Sept. 2003.
[4] See http://loki3.com/flex/triangles.html.
[5] See http://loki3.com/flex/flex/.
[6] For one account, see *The V-flex, Triangle Orientation, and Catalan Numbers in Hexaflexagons* by Ionut E. Iacob, T. Bruce McLean, and Hua Wang, Jan. 2012.
[7] *Flexagons,* "Building and Operating the Flexagon" by Anthony S. Conrad and Daniel K. Hartline, May 1962.
[8] See http://oeis.org/A000108.
[9] *Flexagons,* "Maps and Plans" by Anthony S. Conrad and Daniel K. Hartline, May 1962.
[10] See http://tech.groups.yahoo.com/group/Flexagon_Lovers/message/641.
[11] See http://loki3.com/flex/point-flexagon.html.