Bowers' Named Numbers

Jonathan Bowers has created some giant numbers and given them names on his 'Infinity Scrapers' webpage (www.polytope.net/hedrondude/scrapers.htm). He has made an attempt at creating an 'Extended Array Notation' that goes beyond linear arrays (similar to my Linear Array Notation) on his website but the rules have not been clearly defined. In this document, I will attempt to express some of his numbers using my various array notations.

Linear Arrays

Googol = {10, 100}
  = 10^100,
Googolplex = {10, {10, 100}}
  = 10^10^100 (first two numbers included for comparison),
Decker = {10, 10, 2}
  = 10^^10 (power tower of 10 10's),
Giggol = {10, 100, 2}
  = 10^^100 (power tower of 100 10's),
Giggolplex = {10, {10, 100, 2}, 2}
  = 10^^10^^100,
Tritri = {3, 3, 3} (Tri-tri is Greek for three 3's)
  = 3^^^3,
Gaggol = {10, 100, 3}
  = 10^^^100,
Tridecal = {10, 10, 10} (Tri-decal is Greek for three 10's)
  = 10 {10} 10 (in Extended Operator Notation, {10} is 10 Knuth's up-arrows),
Boogol = {10, 10, 100}
  = 10 {100} 10,
Boogolplex = {10, 10, {10, 10, 100}}
  = 10 {10 {100} 10} 10,
Corporal = {10, 100, 1, 2}
  = 10 {{1}} 100
  = 10 {10 {10 { ... {10 {10} 10} ... } 10} 10} 10 (with 100 10's from centre out),
Biggol = {10, 10, 100, 2}
  = 10 {{100}} 10,
Tetratri = {3, 3, 3, 3}
  = 3 {{{3}}} 3
Baggol = {10, 10, 100, 3}
  = 10 {{{100}}} 10,
General = {10, 10, 10, 10} (also known as Tetradecal)
  = 10 {{{{{{{{{{10}}}}}}}}}} 10,
Troogol = {10, 10, 10, 100},
Pentatri = {3, 3, 3, 3, 3},
Pentadecal = {10, 10, 10, 10, 10},
Quadroogol = {10, 10, 10, 10, 100},
Hexatri = {3, 3, 3, 3, 3, 3},
Hexadecal = {10, 10, 10, 10, 10, 10},
Quintoogol = {10, 10, 10, 10, 10, 100}.

Multi-Dimensional Arrays

Jonathan Bowers used the (n-1) symbol (inside round brackets) to denote my [n] separator in his ‘Extended Array Notation’. The comma was shorthand for (0) in his notation. He writes b^n & a’s to denote b^n array of a’s (or size b n-dimensional array of a’s), sometimes within a larger main array; I use my array bracket string ‘a ‹n› b’ (with the dimensions inside angle brackets) to denote the same array or subarray within a larger array.

Iteral = {10, 10 [2] 2}
= {10, 10, 10, 10, 10, 10, 10, 10, 10, 10},
Goobol = {10, 100 [2] 2}
= {10, 10, 10, ... , 10} (with 100 10’s),
Latri = {3, 3, 3 [2] 2},
Boobol = {10, 10, 100 [2] 2},
Troobol = {10, 10, 10, 100 [2] 2},
Gootrol = {10, 100 [2] 3}
= {10, 10, 10, ... , 10 [2] 2} (with 100 10’s),
Bootrol = {10, 10, 100 [2] 3},
Trootrol = {10, 10, 10, 100 [2] 3},
Emperal = {10, 10 [2] 10}
= {10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 9},
Hyperal = {10, 10 [2] 10, 10}
= {10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 9, 10},
Diteral = {10, 10 [2] 1 [2] 2}
= {10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10},
Admiral = {10, 10 [2] 1 [2] 10}
= {10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 9},

Dutritri = {3, 3 [3] 2}
= {3 ‹2› 3} (Du-tri-tri is Greek for 2 dimensional array of 3^2 3’s)
= {3 ‹1› 3 [2] 3 ‹1› 3 [2] 3 ‹1› 3}
= {3, 3, 3 [2] 3, 3, 3 [2] 3, 3, 3},

Dutridecal = {10, 3 [3] 2}
= {10 ‹2› 3} (Du-tri-decal is Greek for 2 dimensional array of 3^2 10’s)
= {10, 10, 10 [2] 10, 10, 10 [2] 10, 10, 10},

Xappol = {10, 10 [3] 2} = {10 ‹2› 10}
= {10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2]
10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2]
10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2]
10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2]
10, 10, 10, 10, 10, 10, 10, 10, 10, 10 [2] 10, 10, 10, 10, 10, 10, 10, 10, 10, 10},

Dimentri = {3, 3 [4] 2}
= {3 ‹3› 3} (Dimen-tri is all 3’s up to 3 dimensions – 3^3 array of 3’s)
= {3, 3, 3 [2] 3, 3, 3 [2] 3, 3, 3 [3] 3, 3, 3 [2] 3, 3, 3 [2] 3, 3, 3 [3] 3, 3, 3 [2] 3, 3, 3 [2] 3, 3, 3},

Colossol = {10, 10 [4] 2}
= {10 ‹3› 10} (10×10×10 ‘cube’ of 10’s),

Dimendecal = {10, 10 [11] 2}
= {10 ‹10› 10} (Dimen-decal is all 10’s up to 10 dimensions – 10^10 array of 10’s).

Gongulus = {10, 100 [101] 2}
= {10 ‹100› 100} (100 dimensional array of 100^100 10’s),
though Bowers defines this number as a size 10 array,
{10, 10 (100) 2} = {10^100 & 10} (100 dimensional array of 10^100 10’s)
on his ‘Infinity Scrapers’ page.

Hyper-Dimensional Arrays

These are what Jonathan Bowers terms ‘superdimensional’ arrays – the first level of ‘tetrational’ arrays – where an array with n entries inside the separator or angle bracket array is an n superdimensional array. Each of the numbers he uses in his round bracket separators is one less than the corresponding value under my square bracket separator arrays, for example, his (0,1) and ((1)1) symbols are my [1, 2] and [1 [2] 2] respectively.

Dulatri = {3, 3 [1, 3] 2}
= {3 ‹0, 3› 3}
= {3 ‹3, 2› 3} (Du-la-tri – 2 hyperdimensions of 3 dimensions, with three 3’s per row etc.)
= {A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [3,2]
A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [3,2]
A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A},
where A = ‘3,3,3 [2] 3,3,3 [2] 3,3,3 [3] 3,3,3 [2] 3,3,3 [2] 3,3,3 [3] 3,3,3 [2] 3,3,3 [2] 3,3,3’.

Gingulus = {10, 100 [1, 3] 2}
= {10 ‹0, 3› 100}
= {10 ‹100, 2› 100},
2 hyperdimensions of 100 dimensions – a 100^100 array of ‘10 ‹100› 100’s, where ‘10 ‹100› 100’ is a 100^100 array of 10’s.

Trilatri = {3, 3 [1, 4] 2}
= {3 ‹0, 4› 3}
= {3 ‹3, 3› 3} (Tri-la-tri – 3 hyperdimensions of 3 dimensions, with three 3’s per row etc.)
= {B [1,3] B [1,3] B [2,3] B [1,3] B [1,3] B [2,3] B [1,3] B [1,3] B [3,3]
B [1,3] B [1,3] B [2,3] B [1,3] B [1,3] B [2,3] B [1,3] B [1,3] B [3,3]
B [1,3] B [1,3] B [2,3] B [1,3] B [1,3] B [2,3] B [1,3] B [1,3] B},
where B = ‘A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [3,2]
A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [3,2]
A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A [2,2] A [1,2] A [1,2] A’,
where A = ‘3,3,3 [2] 3,3,3 [2] 3,3,3 [3] 3,3,3 [2] 3,3,3 [2] 3,3,3 [3] 3,3,3 [2] 3,3,3 [2] 3,3,3’.

Gangulus = {10, 100 [1, 4] 2}
= {10 ‹0, 4› 100}
= {10 ‹100, 3› 100},

3 hyperdimensions of 100 dimensions – a 100^100 array of ‘10 ‹100, 2› 100’s, where ‘10 ‹100, 2› 100’ is a 100^100 array of ‘10 ‹100› 100’s, where ‘10 ‹100› 100’ is a 100^100 array of 10’s.

Bongulus = {10, 100 [1, 1, 2] 2}
= {10 ‹100, 100› 100},
as Gongulus but with 100 hyperdimensions of 100 dimensions.

Bingulus = {10, 100 [1, 1, 3] 2}
= {10 ‹100, 100, 2› 100},
2 hyper-2 dimensions of 100 hyperdimensions of 100 dimensions.

Bangulus = {10, 100 [1, 1, 4] 2}
= {10 ‹100, 100, 3› 100},
3 hyper-2 dimensions of 100 hyperdimensions of 100 dimensions.

Trimentri = {3, 3 [1, 1, 1, 2] 2}
= {3 ‹3, 3, 3› 3},
all 3’s up to the level of dimensions (hyper-2 is third level of dimensions).

Trongulus = {10, 100 [1, 1, 1, 2] 2}
= {10 ‹100, 100, 100› 100},
100 hyper-2 dimensions of 100 hyperdimensions of 100 dimensions.

Quadrongulus = {10, 100 [1, 1, 1, 1, 2] 2}
= {10 ‹100, 100, 100, 100› 100},
100 hyper-3 dimensions of 100 hyper-2 dimensions of 100 hyperdimensions of 100 dimensions.

Nested Arrays

These are what Jonathan Bowers terms ‘tetrational’ arrays. In this class of arrays, but lying beyond ‘superdimensions’, lies ‘trimensional’ arrays, ‘quadramensional’ arrays and so on. Perhaps ‘bimensional’ arrays may be a better term for ‘superdimensional’ arrays as bi- is a Latin prefix, like the quad- prefix.

Trimentri = {3, 3 [1 [2] 2] 2}
= {3 ‹0 [2] 2› 3}
= {3 ‹3 ‹1› 3› 3}
= {3 ‹3, 3, 3› 3},
all 3’s up to the number of ‘superdimensions’ (or number of levels of the dimensions). Bowers calls this a 1 trimensional array of 3’s, as a 1 dimensional array of three 3’s appears within the outer pair of angle brackets. He also refers to this as a 3 tetrated to 3 array of 3’s (3^^3 array of 3’s). (He could have created a number called Superdimentri or Bimentri, in between Dimentri and Trimentri.)

Goplexulus = {10, 100 [1 [2] 2] 2}
= {10 ‹0 [2] 2› 100}
= {10 ‹100 ‹1› 100› 100}
= {10 ‹100, 100, 100, ... , 100› 100} (with 100 100’s within angle brackets),

this has 100 levels of dimensions; Bowers calls it a 100 superdimensional array.

Goduplexulus = {10, 100 [1 [1, 2] 2] 2}
= {10 ‹0 [1, 2] 2› 100}
= {10 ‹100 ‹0, 2› 100› 100}
= {10 ‹100 ‹100› 100› 100},
I call this a 100 dimensional level 1 nested array; Bowers calls it a 100 trimensional array.

Gotriplexulus = {10, 100 [1 [1 [2] 2] 2] 2}
= {10 ‹0 [1 [2] 2] 2› 100}
= {10 ‹100 ‹0 [2] 2› 100› 100}
= {10 ‹100 ‹100 ‹1› 100› 100› 100}
= {10 ‹100 ‹100, 100, ... , 100› 100› 100} (with 100 100's within inner angle brackets),
this has 100 levels of dimensions within a level 1 nested array; Bowers calls it a 100 quadramensional array.

Goppatoth = {10, 51 [1 \ 2] 2}
= {10 ‹0 \ 2› 51}
= {10 ‹10 ‹10 ‹ ... ‹10 ‹10› 10› ... › 10› 10› 10} (with 50 angle brackets),
I call this a 10 dimensional level 49 nested array; Bowers calls it a {10, 100, 2} array of 10's = 10^^100 array of 10's.

Hyper-Nested Arrays

These are what Jonathan Bowers terms 'pentational' arrays, 'higher operation group' arrays, larger 'array' arrays, right up to 'legion' arrays, 'lugion' arrays, 'lagion' arrays, and beyond! These 'array' arrays are where the 'array' (in inverted commas) is actually a structure in terms of the smallest infinite ordinal ω (though Bowers uses the letter X to represent ω). On the first three pages of Beyond Bird's Nested Arrays I and pages 18-20 of Beyond Bird's Nested Arrays II, I attempted to express the higher ordinals in terms of just ω and my various array notations (up to Hyper-Nested Arrays); this helped me to work out the most probable arrays for the larger finite numbers using my Hyper-Nested notation (e.g. the larger 'hexational' arrays are most likely to correspond to the φ(3, 0) level separator [1 \ 1 \ 1 \ 2]). At this level, my notation is only an approximation of Bowers, so I will be using the approximately equals sign instead of the true equals sign.

Triakulus ≈ {3, 3 [1 \ 1 \ 2] 2}
= {3 ‹0 \ 1 \ 2› 3}
= {3 ‹3 \ 3 ‹3 \ 3› 3› 3},
Bowers' {3, 3, 3} array of 3's = 3^^^3 array of 3's.

Kungulus ≈ {10, 100 [1 \ 1 \ 2] 2}
= {10 ‹0 \ 1 \ 2› 100}
= {10 ‹10\10 ‹10\10 ‹ ... ‹10\10 ‹10\10› 10› ... › 10› 10› 10} (with 99 angle brackets),
Bowers' {10, 100, 3} array of 10's = 10^^^100 array of 10's.

Quadrunculus ≈ {10, 100 [1 \ 1 \ 1 \ 2] 2}
= {10 ‹0 \ 1 \ 1 \ 2› 100}
= {10 ‹10\10\10 ‹10\10\10 ‹ ... ‹10\10\10 ‹10\10\10› 10› ... › 10› 10› 10}

(with 99 angle brackets),
Bowers' {10, 100, 4} array of 10's = 10^^^^100 array of 10's.

Tridecatrix ≈ {10, 10 [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2] 2}
= {10 ‹0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2› 10},
Bowers' {10, 10, 10} array of 10's.

Humongulus ≈ {10, 10 [1 \ 1 \ 1 \ ... \ 1 \ 2] 2} (with 99 backslashes)
= {10 ‹0 \ 1 \ 1 \ ... \ 1 \ 2› 10},
Bowers' {10, 10, 100} array of 10's.

A giant leap in the numbers now.

Golapulus ≈ {10, 100 [1 [1 ¬ 1 [2] 2] 2] 2}
= {10 ‹0 [1 ¬ 1 [2] 2] 2› 100}
= {10 ‹100 ‹100 ¬ 100 ‹1› 100› 100› 100}
= {10 ‹100 ‹100 ¬ 100, 100, ... , 100› 100› 100} (with 100 100's between ¬ and ›),
Bowers' Gongulus array of 10's or (100^100 array of 10's) array of 10's; he defines Gongulus as 10^100 array of 10's.

A truly gigantic leap in the numbers now – far, far greater than before.

Bowers has introduced 'Legion Arrays' (very vaguely defined) on his 'Exploding Array Function' page (www.polytope.net/hedrondude/array.htm). These begin with small linear arrays followed by a legion mark (forward slash) and 2, as in

{b, p / 2} = {b, p-1 / 2} array of b's
= (...(((b array of b's) array of b's) array of b's) ...) array of b's (with p b's).

For example,

{3, 2 / 2} = 3 array of 3's = {3, 3, 3} = 3^^^3,
{3, 3 / 2} = 3^^^3 array of 3's = Triakulus.

The growth rate of his {b, p / 2} is similar to the growth rate of my {b, p [1 [1¬1¬2] 2] 2} array, only that the initial values of {b, p / 2} are smaller, as

{b, p+1 / 2} < {b, p [1 [1¬1¬2] 2] 2} < {b, p+2 / 2} (for p ≥ 2).

Unfortunately, his 'Legion Arrays' (and beyond) are not much use, since the array structures (in terms of ω) 'catches up' with the finite number arrays at θ(Ω^Ω), the large Veblen ordinal. In other words, an array equals its own structure at θ(Ω^Ω), the level of my [1 [1¬1¬2] 2] separator.

Bowers uses a special array notation to describe structures, for example,

X = line,
X^2 = plane,
X^n = n dimensional space,
{X,2,2} = X^X = above all dimensional spaces,
{X,n,2} = tetration array space (n = 2 is superdimensional array, n = 3 is trimensional array),
{X,n,3} = pentation array space,
{X,X,...,X} = linear array space,
L = {b, p / 2} (legion array – above all of the non-legion array notation spaces),

XL = {b, p (/1) 2} (linear legion array),
(X^X)L = {b,p (/0,1) 2} (superdimensional legion array),
{X,3,2}L = {b,p (/(1)1) 2} (trimensional legion array),
{L,2} = {b, p // 2} (legion legion array, {b, p // 2} is {b, p-1 // 2} legion array of b's),
{L,3} = {b, p /// 2} (legion legion legion array),
{L,X} = {b, p (1)/ 2} = {b, p ///.../ 2} (with p legion marks or /'s).

Beyond this, the legion marks would themselves take on an array structure (not just the separate legions), for example,

{L,{X,3,2}} = {b, p ((1)1)/ 2} ({b, 2 ((1)1)/ 2} = {b, 2 /(1)//(0,1)//(1)//(1,1)//(1)//(0,1)//(1)// 2}).

This is followed by

{L,2,2} = {L,L} (legion marks taking on legion space),
{L,3,2} = {L,{L,2,2}} = {L,{L,L}} (legion marks taking on a legion of legions).

I have attempted to link Bowers' structural array notation to the separators used in my Hyper-Nested Arrays. A selection of them are shown below:-

Structural array	Separator [S] in array {b, p [S] 2}	
L	[1 [1¬1¬2] 2]	(equivalent to Bowers' / legion mark)
XL	[2 [1¬1¬2] 2]	(equivalent to Bowers' (/1) mark)
(X^X)L	[1,2 [1¬1¬2] 2]	(equivalent to Bowers' (/0,1) mark)
{X,X,2}L	[1 [1 \ 2] 2 [1¬1¬2] 2]	
{L,2}	[1 \ 2 [1¬1¬2] 2]	(equivalent to Bowers' // mark)
{L,3}	[1 \ 3 [1¬1¬2] 2]	(equivalent to Bowers' /// mark)
{L,X}	[1 \ 1,2 [1¬1¬2] 2]	(equivalent to Bowers' (1)/ mark)
{L,X^2}	[1 \ 1,1,2 [1¬1¬2] 2]	(equivalent to Bowers' (2)/ mark)
{L,X^X}	[1 \ 1 [2] 2 [1¬1¬2] 2]	(equivalent to Bowers' (0,1)/ mark)
{L,{X,X,2}}	[1 \ 1 [1 \ 2] 2 [1¬1¬2] 2]	
{L,2,2} = {L,L}	[1 \ 1 [1 [1¬1¬2] 2] 2 [1¬1¬2] 2]	
{L,3,2} = {L,{L,2,2}}		
= {L,{L,L}}	[1 \ 1 [1 \ 1 [1 [1¬1¬2] 2] 2 [1¬1¬2] 2] 2 [1¬1¬2] 2]	
{L,X,2}	[1 \ 1 \ 2 [1¬1¬2] 2]	
{L,X+1,2}	[1 \ 2 \ 2 [1¬1¬2] 2]	
{L,2X,2}	[1 \ 1,2 \ 2 [1¬1¬2] 2]	
{L,2,3} = {L,L,2}	[1 \ 1 [1 [1¬1¬2] 2] 2 \ 2 [1¬1¬2] 2]	
{L,X,3}	[1 \ 1 \ 3 [1¬1¬2] 2]	
{L,X,X}	[1 \ 1 \ 1,2 [1¬1¬2] 2]	
{L,2,1,2} = {L,L,L}	[1 \ 1 \ 1 [1 [1¬1¬2] 2] 2 [1¬1¬2] 2]	
{L,X,1,2}	[1 \ 1 \ 1 \ 2 [1¬1¬2] 2]	
{L,X,1,3}	[1 \ 1 \ 1 \ 3 [1¬1¬2] 2]	
{L,X,1,1,2}	[1 \ 1 \ 1 \ 1 \ 2 [1¬1¬2] 2]	
{L,X (1) 2}	[1 [2¬2] 2 [1¬1¬2] 2]	
{L,X (1) 3}	[1 [2¬2] 3 [1¬1¬2] 2]	
{L,X (1) X}	[1 [2¬2] 1,2 [1¬1¬2] 2]	
{L,2 (1) 1,2}		
= {L,L,...,L (1) L}	[1 [2¬2] 1 [1 [1¬1¬2] 2] 2 [1¬1¬2] 2]	
{L,X (1) 1,2}	[1 [2¬2] 1 \ 2 [1¬1¬2] 2]	
{L,X (2) 2}	[1 [3¬2] 2 [1¬1¬2] 2]	

Beyond this, we get arrays on the legion space (e.g. {L,100 (100) 2} is a 100^100 legiattic array of L's), and repeated 'legiattic array of L's' leads to 'Lugion Arrays' (represented by L2).

L2 = {b, p \ 2}   (lugion array, {b, p \ 2} is {b, p-1 \ 2} legiattic array of b's),
{L2,2} = {b, p \\ 2}   (lugion lugion array, {b, p \\ 2} is {b, p-1 \\ 2} lugion array of b's),
{L2,X} = {b, p (1)\ 2} = {b, p \\\...\ 2}   (with p lugion marks or \'s),
{L2,L}   (lugion marks taking on legion space),
{L2,2,2} = {L2,L2}   (lugion marks taking on lugion space).

Beyond that, we get arrays on the lugion space (e.g. {L2,100 (100) 2} is a 100^100 lugiattic array of L2's), and repeated 'lugiattic array of L2's' leads to 'Lagion Arrays' (represented by L3). Repeated 'lagiattic array of L3's' leads to 'Ligion Arrays' (represented by L4), and so on.

L3 = {b, p | 2}   (lagion array, {b, p | 2} is {b, p-1 | 2} lugiattic array of b's),
L4 = {b, p - 2}   (ligion array, {b, p - 2} is {b, p-1 - 2} lagiattic array of b's).

Continuing with linking Bowers' structural array notation to the separators used in my Hyper-Nested Arrays, some more of them are shown below:-

Structural array	Separator [S] in array {b, p [S] 2}	
L2	[1 [1¬1¬2] 3]	(equivalent to Bowers' \ lugion mark)
{L2,2}	[1 \ 2 [1¬1¬2] 3]	(equivalent to Bowers' \\ mark)
{L2,X}	[1 \ 1,2 [1¬1¬2] 3]	(equivalent to Bowers' (1)\ mark)
{L2,L}	[1 \ 1 [1 [1¬1¬2] 2] 2 [1¬1¬2] 3]	
{L2,2,2} = {L2,L2}	[1 \ 1 [1 [1¬1¬2] 3] 2 [1¬1¬2] 3]	
L3	[1 [1¬1¬2] 4]	(equivalent to Bowers' | lagion mark)
L4	[1 [1¬1¬2] 5]	(equivalent to Bowers' - ligion mark)
L100	[1 [1¬1¬2] 101]	
LX	[1 [1¬1¬2] 1,2]	
L(X+1)	[1 [1¬1¬2] 2,2]	
L(2X)	[1 [1¬1¬2] 1,3]	
L(X^2)	[1 [1¬1¬2] 1,1,2]	
L(X^X)	[1 [1¬1¬2] 1 [2] 2]	
L{X,X,2}	[1 [1¬1¬2] 1 [1 \ 2] 2]	
LL	[1 [1¬1¬2] 1 [1 [1¬1¬2] 2] 2]	
LL2	[1 [1¬1¬2] 1 [1 [1¬1¬2] 3] 2]	
LLL	[1 [1¬1¬2] 1 [1 [1¬1¬2] 1 [1 [1¬1¬2] 2] 2] 2]	
X (line) of L's = (1)L	[1 [1¬1¬2] 1 \ 2]	
X^2 of L's = (2)L	[1 [1¬1¬2] 1 [2¬2] 2]	
X^X of L's = (0,1)L	[1 [1¬1¬2] 1 [1,2¬2] 2]	
{X,X,2} of L's	[1 [1¬1¬2] 1 [1 \ 2 ¬2] 2]	
L array of L's	[1 [1¬1¬2] 1 [1¬1¬2] 2]	
L100 array of L's	[1 [1¬1¬2] 1 [1¬1¬2] 101]	

The links between Bowers' structural arrays and my Hyper-Nested Arrays enables me to make approximations to his largest named numbers using my Hyper-Nested notation. These are as follows:-

Big Boowa ≈ {3, 3, 3 [1 [1¬1¬2] 2] 2},

Grand Boowa ≈ {3, 3, {3,3,3 [1 [1¬1¬2] 2] 2} [1 [1¬1¬2] 2] 2},

Super Gongulus ≈ {10, 100 [101] 2 [1 [1¬1¬2] 2] 2}
= {10 ‹100› 100 [1 [1¬1¬2] 2] 2}
(if Gongulus is {10 ‹100› 100} rather than {10 ‹100› 10}),

Wompogulus ≈ {10, 10 [11] 2 [1 [1¬1¬2] 2] 100}
= {10 ‹10› 10 [1 [1¬1¬2] 2] 100}.

Guapamonga ≈ {10, 10 [101 [1¬1¬2] 2] 2}
= {10 ‹100 [1¬1¬2] 2› 10},
Bowers' (X^100)L structural array – 10^100 legion array of A's, where A is a 10^100 array of 10's, as shown here: {A/A/A/A/A/A/A/A/A/A(/2)A/A/A/A/A/A/A/A/A/A(/2)...(/99)...A/A}, where A = 10^100 & 10's.

Big Hoss ≈ {100, 100 [1 \ 1,2 [1¬1¬2] 2] 2}
= {100 ‹0 \ 1,2 [1¬1¬2] 2› 100}
= {100 ‹100 \ 100 [1¬1¬2] 2› 100},
Bowers' {L,X} structural array (line of 100 legion marks).

Bukuwaha ≈ {100, 100 [1 \ 1 [2] 2 [1¬1¬2] 2] 2}
= {100 ‹0 \ 1 [2] 2 [1¬1¬2] 2› 100}
= {100 ‹100 \ 100 ‹1› 100 [1¬1¬2] 2› 100}
= {100 ‹100 \ 100,100,...,100 [1¬1¬2] 2› 100}   (100 100's between \ and [1¬1¬2]),
Bowers' {L,X^100} structural array (100^100 array of legion marks).

Goshomity ≈ {100, 100 [1 \ 1,2 [1¬1¬2] 3] 2}
= {100 ‹0 \ 1,2 [1¬1¬2] 3› 100}
= {100 ‹100 \ 100 [1¬1¬2] 3› 100},
Bowers' {L2,X} structural array (line of 100 lugion marks).

Big Bukuwaha ≈ {100, 100 [1 \ 1 [1 \ 1 [2] 2 [1¬1¬2] 2] 2 [1¬1¬2] 3] 2}
= {100 ‹0 \ 1 [1 \ 1 [2] 2 [1¬1¬2] 2] 2 [1¬1¬2] 3› 100}
= {100 ‹100 \ 100 ‹0 \ 1 [2] 2 [1¬1¬2] 2› 100 [1¬1¬2] 3› 100}
= {100 ‹100 \ 100 ‹100 \ 100 ‹1› 100 [1¬1¬2] 2› 100 [1¬1¬2] 3› 100}
= {100 ‹100 \ 100 ‹100 \ 100,100,...,100 [1¬1¬2] 2› 100 [1¬1¬2] 3› 100}
(with 100 100's in 100,100,...,100),
Bowers' {L2,{L,X^100}} structural array (Bukuwaha array of lugion marks).

Bongo Bukuwaha
≈ {100, 100 [1 \ 1 [1 \ 1 [1 \ 1 [2] 2 [1¬1¬2] 2] 2 [1¬1¬2] 3] 2 [1¬1¬2] 4] 2}
= {100 ‹100 \ 100 ‹100 \ 100 ‹100 \ 100,100,...,100 [1¬1¬2] 2› 100 [1¬1¬2] 3› 100 [1¬1¬2] 4› 100}
(with 100 100's in 100,100,...,100),
Bowers' {L3,{L2,{L,X^100}}} structural array (Big Bukuwaha array of lagion marks).

Quabinga Bukuwaha
≈ {100, 100 [1 \ 1 [1 \ 1 [1 \ 1 [1 \ 1 [2] 2 [1¬1¬2] 2] 2 [1¬1¬2] 3] 2 [1¬1¬2] 4] 2 [1¬1¬2] 5] 2}
= {100 ‹100 \ 100 ‹100 \ 100 ‹100 \ 100 ‹100 \ 100,100,...,100 [1¬1¬2] 2› 100 [1¬1¬2] 3› 100 [1¬1¬2] 4› 100 [1¬1¬2] 5› 100}
(with 100 100's in 100,100,...,100),
Bowers' {L4,{L3,{L2,{L,X^100}}}} structural array (Bongo Bukuwaha array of ligion marks).

Meameamealokkapoowa ≈ {10, 10 [1 \ 10 [1¬1¬2] 101] 2}
= {10 ‹0 \ 10 [1¬1¬2] 101› 10},
Bowers' {L100,10} structural array.

Meameamealokkapoowa Oompa ≈ {10, 10 [1 \ 10 [1¬1¬2] 1 \ 10 [1¬1¬2] 101] 2}
= {10 ‹0 \ 10 [1¬1¬2] 1 \ 10 [1¬1¬2] 101› 10},
Bowers' {LLL...L,10} structural array, where LLL...L is a {L100,10} array of L's.

My Hyper-Nested Array Notation is so powerful that Bowers' largest named number (Meameamealokkapoowa Oompa) does not even get as far as

{3, 4 [1 [2¬1¬2] 2] 2} = {3 ‹0 [2¬1¬2] 2› 4}
= {3 ‹4 [1¬1¬2] 4 [1¬1¬2] 4 [1¬1¬2] 4› 4},

using my notation. And I have gone much, much further than this by inventing my Nested Hyper-Nested Array Notation, which is shown in Beyond Bird's Nested Arrays III. The ¬ sign in that document is the 2-hyperseparator backslash ($\backslash_2$), which is just the second in the sequence of many special hypersymbols, the nth such symbol being the n-hyperseparator backslash ($\backslash_n$).

The H(n) function at the end of Beyond Bird's Nested Arrays III is in a completely different universe of numbers!

Author: Chris Bird (Gloucestershire, England, UK)
Last modified: 19 March 2012
E-mail: m.bird44 at btinternet.com (not clickable to thwart spambots!)