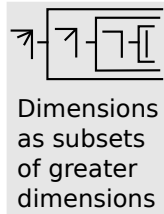
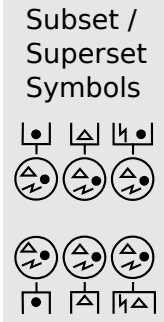
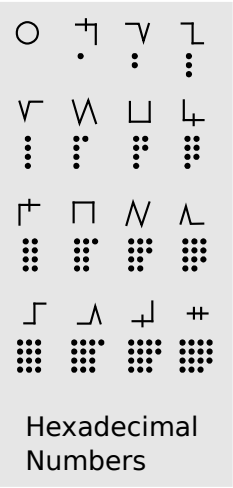
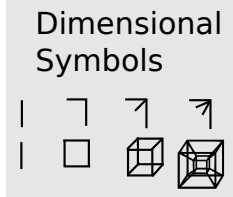
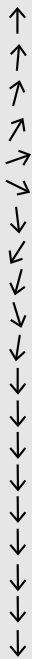


Arrow of Direction

This pattern shows an arrow achieving stability in the direction the arrow points

Laws of fluid dynamics make this universally true (gases and liquids are both fluids)

This also helps establish top to bottom as the direction of reading



Time / Entropy symbols

Defined via common universal examples of time progressing and entropy increasing

Orbital movement occurs on many levels
Movement arrow defines orbital direction
Orbital progression implies time steps

Particle decay into waves and particle cluster dispersion both imply time and entropy both progress

Wave elongating (redshifting) implies both time and energy progression

Elastic collision implies time progression

Reading direction is top → bottom, so we will standardize everything flows/increases top → bottom. Time, value, even the binary numbers will use this single standard.

There is no reason why top/up must mean increasing value, that is just cultural convention in charts.

We effectively use 2 conflicting standards top → bottom for reading (first line, second line, third line increase downwards) and bottom → top for data and charts. These are arbitrary.

Some people instinctively count things in their visual space upwards, others downwards. There are “arguments” for either way, but none are related to any universal law or logic.

The best argument I have ever heard is :

“if you make a Cartesian map and label values increasing upwards, then each step forward represents an increase in value upwards on the map and you are oriented correctly (left right in life are left right on the map).”

This argument is still arbitrary for 2 reasons

1. Each step could also be seen as -1 instead
2. It only works “Map North/Up”, if you walk “south/down” then it fails.

In writing everything flows downwards, this is also arbitrary, but it has a good reasons:

1. High up things can be seen from further so the most important information should be at the top. Eg. if you have several labels or flags the most important should be highest up so it can be seen from furthest away and be less likely to be blocked.
2. If you are human, with human arm and hand joints, using wet ink or charcoal/lead/graphite/etc., after you write a line of text you must move downwards, otherwise you wrist and arm could smudge the still wet ink.

Universally, it is all arbitrary, so we choose increase/flow top → bottom for everything
You could rotate the page 180 degrees, and read bottom → top, right → left if you like.

←↑↖↗↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿

Symbol structure for scientific notation

Arrow attached to a number symbol on either side

This line only establishes symbol structure, definition in next lines

↖↙:↖↙↘↙ a←1=a*(10^1)	↖↘:↖↙↘↙ a←B=a*(10^B)	↖↙:↖↙↘↙↘↙↘↙ 5.9←4=59000
-------------------------	-------------------------	----------------------------

↘↙:↘↙↖↙↖↙ a→1=a/(10^1)	↘↙:↘↙↖↙↖↙ a→5=a/(10^5)	↖↙↘↙↘↙:↖↙↘↙↖↙↖↙ 67C2→3=6.7C2
---------------------------	---------------------------	---------------------------------

Scientific notation defined

↖↙↘↙:↖↙↘↙:↖↙↘↙:↖↙↘↙:↖↙↘↙ 1/2=1/2=1/2=1/2=1/2	↖↙↘↙:↖↙↘↙:↖↙↘↙:↖↙↘↙ 1/3=1/3=1/3=1/3	↖↙↘↙:↖↙↘↙:↖↙↘↙:↖↙↘↙ 2/3=2/3=2/3=2/3
-------------------------------------------------	----------------------------------------	----------------------------------------

↖↙↘↙:↖↙↘↙:↖↙↘↙	↖↙↘↙:↖↙↘↙:↖↙↘↙	↖↙↘↙:↖↙↘↙:↖↙↘↙	↖↙↘↙:↖↙↘↙
1/15	7/9	4/8=4/8=1/2=0.8	1/4←3=400

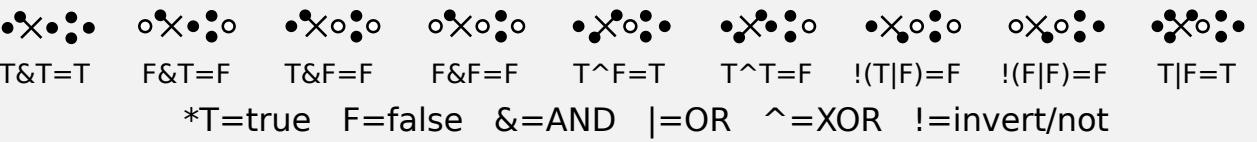
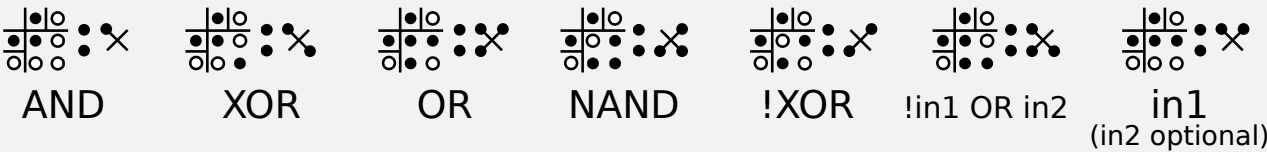
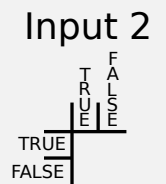
Fraction symbols defined



↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿


↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	0	As with all number symbols if all values are on one level then they are all 1's
0001 0100 0111 1000 0010 1001 1101 10000 1111	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	1	
1 4 7 10 2 9 D 10 F			

↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	↖↙↘↙↞↠↡↢↣↤↥↦↧↨↩↪↫↬↭↮↯↰↱↲↳↴↵↶↷↸↹↺↻↼↽↾↿⇀⇁⇂⇃⇄⇅⇆⇇⇈⇉⇊⇋⇌⇍⇎⇏⇐⇑⇒⇓⇔⇕⇖⇗⇘⇙⇚⇛⇜⇝⇞⇟⇠⇡⇢⇣⇤⇥⇦⇧⇨⇩⇪⇫⇬⇭⇮⇯⇰⇱⇲⇳⇴⇵⇶⇷⇸⇹⇺⇻⇼⇽⇾⇿	0
0001/0010	01/0010	1/0010	1/10	1/0011	1/11	1/100
1/2	1/2	1/2	1/2	1/3	1/3	1/4











- = zero / FALSE / space
 - = TRUE / particle
- The space/particle meanings are defined in previous and later sections
In logic gates we only use the TRUE / FALSE meaning





 Uscrip numbers use a binary representation system with 0/FALSE on top and 1/TRUE below
 Values increase downwards for all other representaions as well












 The invert / negative symbol shows that pattern reversed
 True/value above zero/space

*The symbolism does not need to be known, meaning defined below






















!T=F !F=T 0-1=-1 3-9=-6 -3-4=-7 3-(-4)=7 -3-(-4)=1







*invert represented here using !(exclamation mark)







2*-4=-8 -2*-2=4 -6/2=-3 6/-2=-3 -6/-2=3











$(a^{-b}) = (1/(a^b))$







$-2^3 = -2*2*2 = -8$ $-2^2 = 4$















$r1=1$ $-r1=-1$ $\sqrt[2]{1}=r1$ $\sqrt[2]{-1}=i1$
 r=real number i=imaginary number

$(\sqrt[b]{a}) = (1/(\sqrt[b]{a}))$















$\sqrt[3]{-9} = -3$ $\sqrt[2]{-16} = i4$ $i2^4 = 16$ $--1 = 1$

$abs(a) = \sqrt[2]{a^2}$

This symbol is defined as a formula which returns the absolute value of a number

Above you see the various elements defined so far in a sequence separated by the time progression symbol. This shows the order of operations, akin to the BEDMAS sequence.

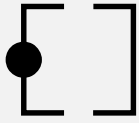















$1 / 16 * 4 - 2 = ((1 / 16) * 4) - 2 = (1/16 * 4) - 2 = \frac{1}{2} - 2 = -3/2 = -1.8 = -1.8$

An example showing operation sequence and various other defined structures. How many examples are "enough" is a matter of debate.

We can add as many examples as we like, but I feel there is already enough to be decoded easily, if you disagree just add more examples.

Evaluation Bracket

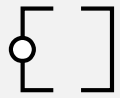


The evaluation bracket reduces anything to a True or False value.
It is a bracket with a particle dot, representing that it reduces to a particle
Any value except 0 is True
It turns statements into comparisons

eg. equals statement (=) turns into a comparison (==)

$\bullet [1] \bullet \bullet$	$\bullet [2] \bullet \circ$	$\bullet [\neg] \bullet \bullet$	$\bullet [\neq] \bullet \bullet$	$\bullet [1 \neq 1] \bullet \bullet$	$\bullet [1 \neq 0] \bullet \circ$	$\bullet [0 \neq 0] \bullet \bullet$
eval(1)=T	eval(2)=T	eval(-1)=T	eval(1==0)=F			
	eval(2)=F	eval(1/3)=T	eval(1==1)=T		eval(0==0)=T	

Inverted Evaluation Bracket



Works the same as Evaluation
Just returns the opposite value

Not-Equals Symbol

Works the same as Equals \bullet
Just returns opposite value \circ

$\bullet [a] \bullet \bullet$	$\circ [a] \bullet \bullet$	$\bullet [0] \bullet \bullet$	$\bullet [a * \text{True}] \bullet \bullet$	$\bullet [a * \text{False}] \bullet \circ$	$\bullet [a = b] \bullet \bullet$	$\circ [a = b] \bullet \bullet$
! eval(a)=!eval(a)	!eval(0)=T		a*True=a	a*False=0	eval(a=b)=!eval(a!=b)	
	!eval(a)=F	!eval(0)=T	a*True=a			

Multiplication of X by TRUE is X, and by FALSE is 0
eg "x = a + (b * eval(should I add b to a?))"

Absolute equals / magnitude equals

Same as "=" except positive and negative values are considered the same value

$\bullet [a \text{ abs} = b] \bullet \bullet$	$\bullet [3 \text{ abs} = 3] \bullet \bullet$	$\bullet [4 \text{ abs} = 3] \bullet \circ$	$\bullet [3 \text{ abs} = -3] \bullet \bullet$
(a abs= b) = (abs(a) = abs(b))	eval(-3 abs= 3)=T	eval(4 abs= 3)=F	eval(3 abs= -3)=T

Greater than >

Less than <


$\bullet [a \bullet \circ b] \bullet \bullet$	$\bullet [a \bullet \bullet b] \bullet \bullet$
-----------------------------------------------	-------------------------------------------------

$$(a > b) = (1/2 * ((a - b) + \text{abs}(a - b)))$$


$$(a < b) = (1/2 * ((b - a) + \text{abs}(b - a)))$$

$\bullet [1 \bullet \circ 1] \bullet \circ$	$\bullet [2 \bullet \circ 1] \bullet \bullet$	$\bullet [4.5 \bullet \circ 3] \bullet \bullet$	$\bullet [1 \bullet \circ 2] \bullet \circ$	$\bullet [-1 \bullet \circ 1] \bullet \circ$	$\bullet [1 \bullet \circ -1] \bullet \bullet$
1 > 1 = 0	2 > 1 = 1	4.5 > 3 = 1.5	1 > 2 = 0	-1 > 1 = 0	1 > -1 = 2

These >, < don't just return True or False, if True they return the absolute difference in value
Since zero is False and any value is True, this can be used as a comparator

 "is greater than"

 Range


"is less than" 

$$(a \text{ is} > b) = (\text{eval}(a > b) = T)$$

$$((a) \text{ is} > (b - c)) \text{ range } c$$

$$((a) \text{ is} > (b - c) \ \& \ (a \text{ is} < (b + c)))$$

"is greater/less than" can create statements and definitions
Range allows tolerance on values

 \approx approximately equal to

$$(a \approx 1.2) = ((a) \text{ is} \text{sub_of } 1.2 \text{ range } 0.08)$$

$$(a \approx 1) = ((a) \text{ is} \text{sub_of } 1 \text{ range } 0.8)$$

$$\sqrt{2} \approx 1.6A$$

$$\sqrt[3]{2} \approx 1.6A$$

$$(a \approx 2 * 10^3) = (a = (2 * 10^3) \text{ range } (0.8 * 10^3))$$

$$\sqrt{2} \approx 1.6A0A$$

$$\sqrt[3]{3} \approx 0.8B$$

$$\sqrt[3]{2} \approx 1.6A0A$$

$$\sqrt[3]{3} \approx 0.8B$$

$$\sqrt[3]{3} \approx 0.5$$

"approx equal to" is defined as a rounded number

Default Operation : Multiplication

Fraction Variable
= Fraction * Variable

Fraction Number
= Fraction * Number

$$\text{Fraction sub} \left(\frac{1}{2} \ \frac{1}{3} \ \frac{1}{4} \ \frac{2}{3} \right)$$

$$\text{Fraction Fraction}$$

$$\text{Variable Variable}$$

$$\text{Variable Variable}$$

$$\frac{1}{4} \frac{1}{2} 10 = \frac{1}{4} * \frac{1}{2} * 10 = 2$$

$$\frac{1}{4} \frac{1}{2} / 10 = \frac{1}{4} * (\frac{1}{2} / 10) = 2 * 10^{-2}$$


$$ab = a * b$$

$$\frac{1}{4} \frac{1}{2} 10 = \frac{1}{4} * \frac{1}{2} * 10 = 2$$

$$\frac{1}{4} \frac{1}{2} / 10 = \frac{1}{4} * (\frac{1}{2} / 10) = 2 * 10^{-2}$$


$$ab = a * b$$

Here we also create a symbol for fraction, so we now have

 variable

 real number

 fraction

 imaginary number

Rectangle / Cuboid space

cuboid volume
 $= \text{cub } x * \text{cub } y * \text{cub } z$

surface of cuboid
 $= 2(xy+xz+yz)$



Rectangle area
 $= \text{rect } x * \text{rect } y$

perimeter of rectangle
 $= 2xy$

The perimeter of a rectangle is defined as "1D line sub(rectangle)" or "1D line contains 2D rectangle".

The surface of a cuboid is defined as "2D space that contains the 3D cuboid"

The equations define these terms, they don't need to be inferred from the symbols

We don't do it in reverse (eg. surface is the 2D sub of 3D) because higher dimensions can contain infinite sub-spaces, a term used later

Circles, π , & e



The symbol for "circle area" contains a second circle inside, this is because this represents a "perfect circle" and without it is just "general area/2D space"

This distinction will be come later, we don't need to define it yet.

For now we just need to use the same symbol we use later for "perfect circle"

Spheres & Triangles



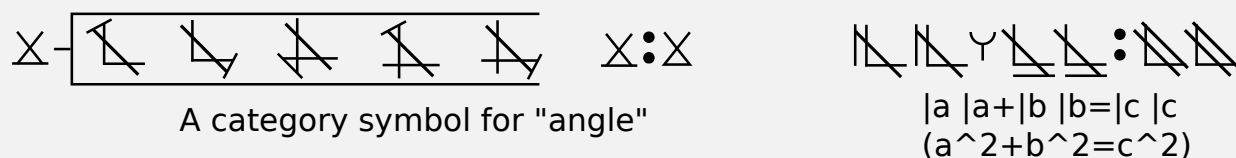
We have reused the circumference symbol

in the context of circles it is circumference, in the context of triangle or angles it is 360°

If you really don't like this and want to avoid ambiguity

you can use the same structure as perimeter for circumference "1D line contains 2D circle"

Basic Trig



$\sin(\angle a) = |b| / |a|$

$\cos(\angle a) = |a| / |a|$

$\tan(\angle a) = |b| / |a|$

Sum() & Count()

$$\left[\left[\begin{array}{c} \text{a} \\ \text{b} \\ \text{c} \\ \text{d} \end{array} \right] \right] : \text{a} + \text{b} + \text{c} + \text{d}$$

$$\text{sum(a,b,c,d) = a+b+c+d}$$

$$\left[\left[\begin{array}{c} 1 \\ 2 \\ 9 \\ 6 \end{array} \right] \right] : 1 + 2 + 9 + 6 = 12$$

$$\text{sum(1,2,9,6) = 12}$$

$$\left[\left[\begin{array}{c} \text{a} \\ \text{b} \\ \text{c} \\ \text{d} \end{array} \right] \right] : 1 + 1 + 1 + 1 = 4$$

$$\text{count(a,b,c,d) = 1+1+1+1=4}$$

$$\left[\left[\begin{array}{c} 1 \\ 9 \end{array} \right] \right] : 2$$

$$\text{count(1,9)=2}$$

Here we introduce the structure for arrays
 Arrays are encapsulated in normal brackets
 Array element brackets are rotated 90 degrees.

Array Variables, Cases & Average

$$\left[\left[\begin{array}{c} \text{a} \\ \text{b} \\ \text{c} \end{array} \right] \right] : \left(\left[\begin{array}{c} \text{sum(a)} \\ \text{count(a)} \end{array} \right] \right)$$

$$(a=(3,5,C)) \text{ sub(sum(a)=14 count(a)=3)}$$

$$\left[\left[\begin{array}{c} \text{a} \end{array} \right] \right] \wedge \left(\left[\begin{array}{c} \text{sum(a)} \\ \text{count(a)} \end{array} \right] \right) : \left(\left[\begin{array}{c} \text{avg(a)} \end{array} \right] \right)$$

$$\text{sum(a)/count(a) = average(a)}$$

above you see a "case"

in brackets we define the case (a variable contains an array)

in a sub of the brackets we can discuss the case

Array Math

$$\left[\left[\begin{array}{c} \text{a} \\ \text{b} \\ \text{c} \end{array} \right] \right] : \left(\left[\begin{array}{c} \text{a}^2 \\ \text{a}^2 \\ \text{a}^2 \end{array} \right] \right) : \left[\begin{array}{c} \text{avg(a^2)} \\ \text{avg(a)} \end{array} \right]$$

$$(a=(1,9,2)) \text{ sub(a^2 = (1^2 , 9^2 , 2^2) = (1,51,4)}$$

Here we define that using math operations on an array will apply that operation to all the elements of the array

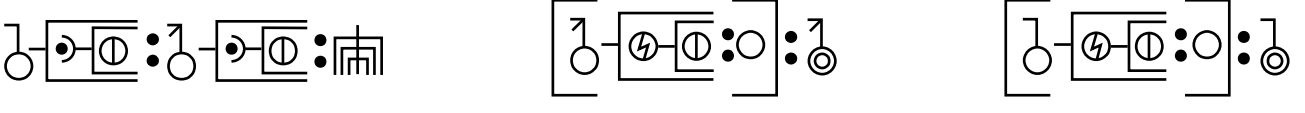
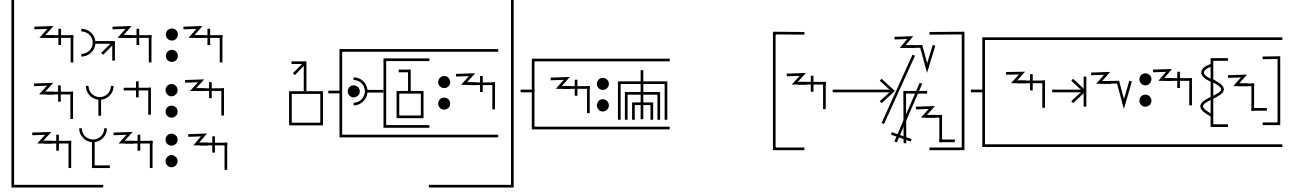
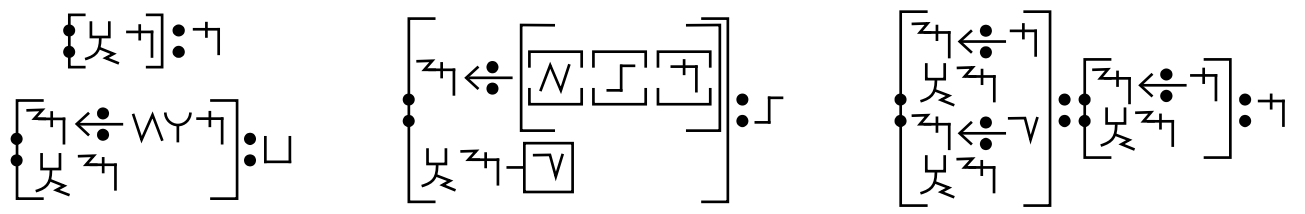
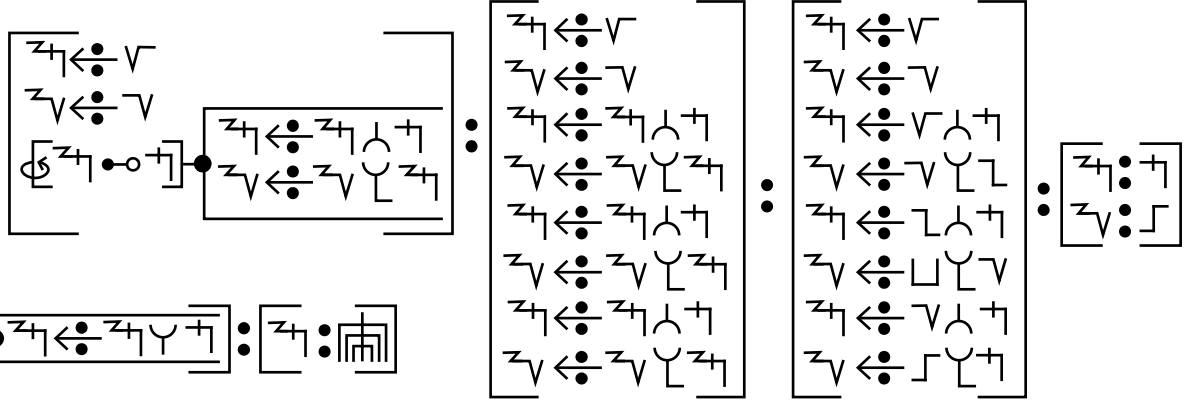
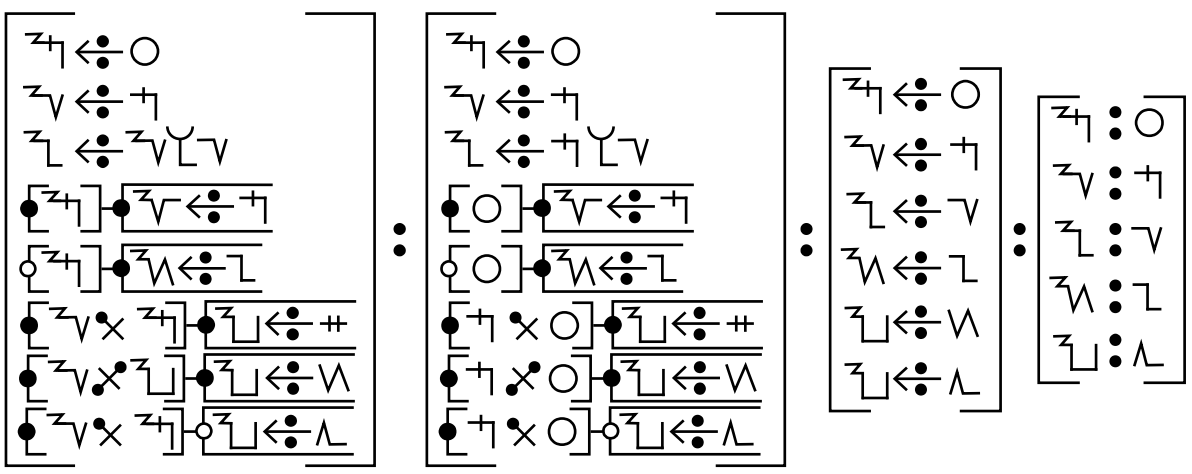
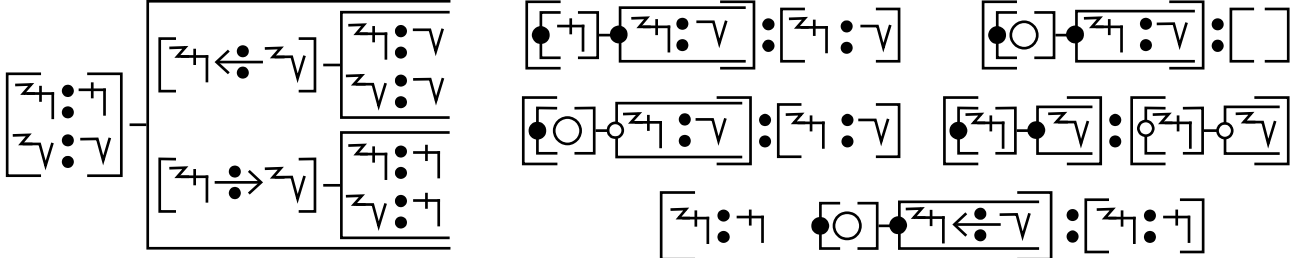
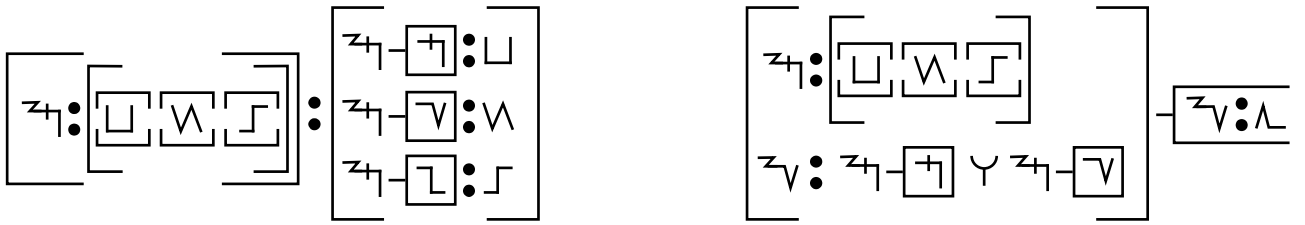
Standard deviation

$$\left(\left[\begin{array}{c} \text{a} \end{array} \right] \right) : \left(\left[\begin{array}{c} \text{avg(a^2)} \\ \text{avg(a)} \end{array} \right] \right) \wedge \left(\left[\begin{array}{c} \text{a} \end{array} \right] \right) \wedge \left(\left[\begin{array}{c} \text{a} \end{array} \right] \right)$$

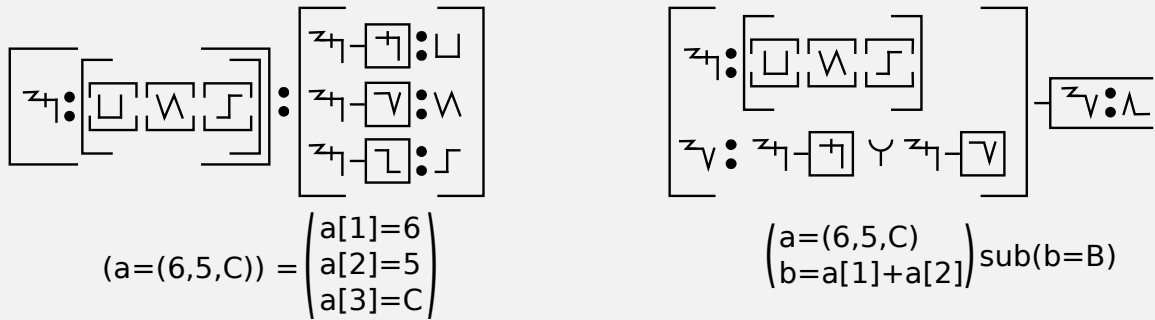
$$\sigma(a) = \sqrt{(\text{avg}(a^2) - (\text{avg}(a)^2))}$$

σ (sigma) represents standard deviation

This is not the standard format for the standard deviation formula, but it is equivalent and works just as well

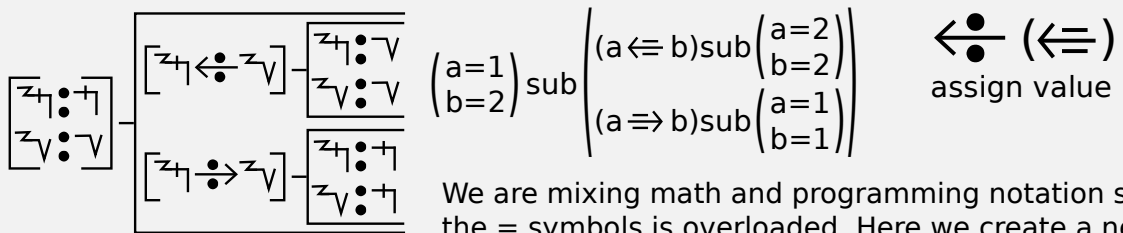


Array element reference



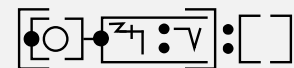
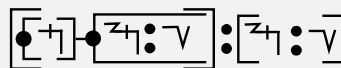
This establishes a format for assign and reference array elements individually. The "array element" ([] square bracket) is a closed box "sub" symbol

Value assignment & Conditionals



We are mixing math and programming notation so the = symbols is overloaded. Here we create a new symbol to distinguish usage types.

Is the condition True?



$$(if(1)\{a=2\}) = (a=2)$$

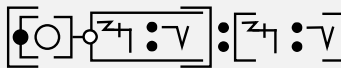
$$(if(0)\{a=2\}) = ()$$

Is the condition False?



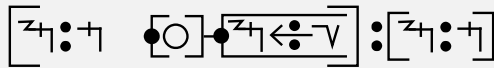
$$(if(0)\{\}\text{else}\{a=2\}) = (a=2)$$

Is the condition True?

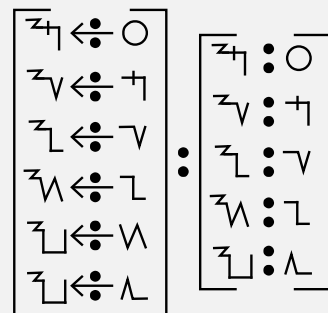
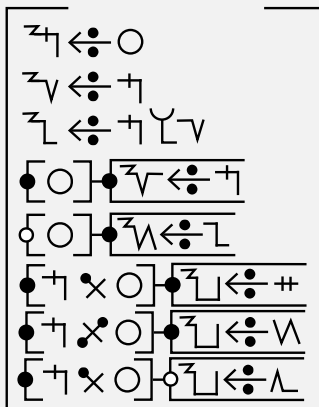
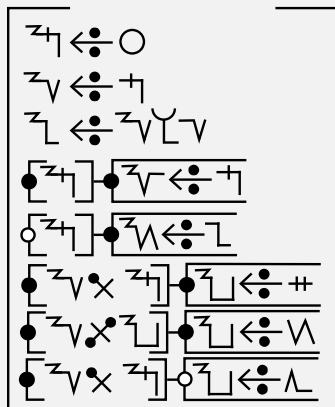


$$(if(a)\{b\}) = (if-not(a)\{\}\text{else}\{b\})$$

Is the condition False?



$$(a=1 \text{ if}(0)\{a <=> 2\}) = (a=1)$$

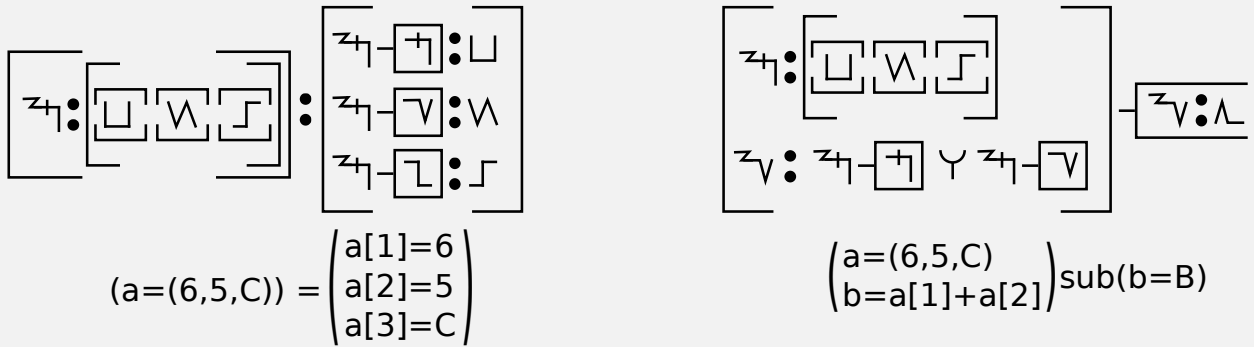


$$\left(\begin{array}{l} a <=> 0 \\ b <=> 1 \\ c <=> b*2 \\ if(a)\{d <=> 1\} \\ if-not(a)\{e <=> 3\} \\ if(b \text{ AND } a)\{f <=> F\} \\ if(b \text{ XOR } a)\{f <=> 5\} \\ if(b \text{ AND } a)\{\}\text{else}\{f <=> B\} \end{array} \right)$$

$$\left(\begin{array}{l} a <=> 0 \\ b <=> 1 \\ c <=> 1*2 \\ if(0)\{d <=> 1\} \\ if-not(0)\{e <=> 3\} \\ if(1 \text{ AND } 0)\{f <=> F\} \\ if(1 \text{ XOR } 0)\{f <=> 5\} \\ if(1 \text{ AND } 0)\{\}\text{else}\{f <=> B\} \end{array} \right)$$

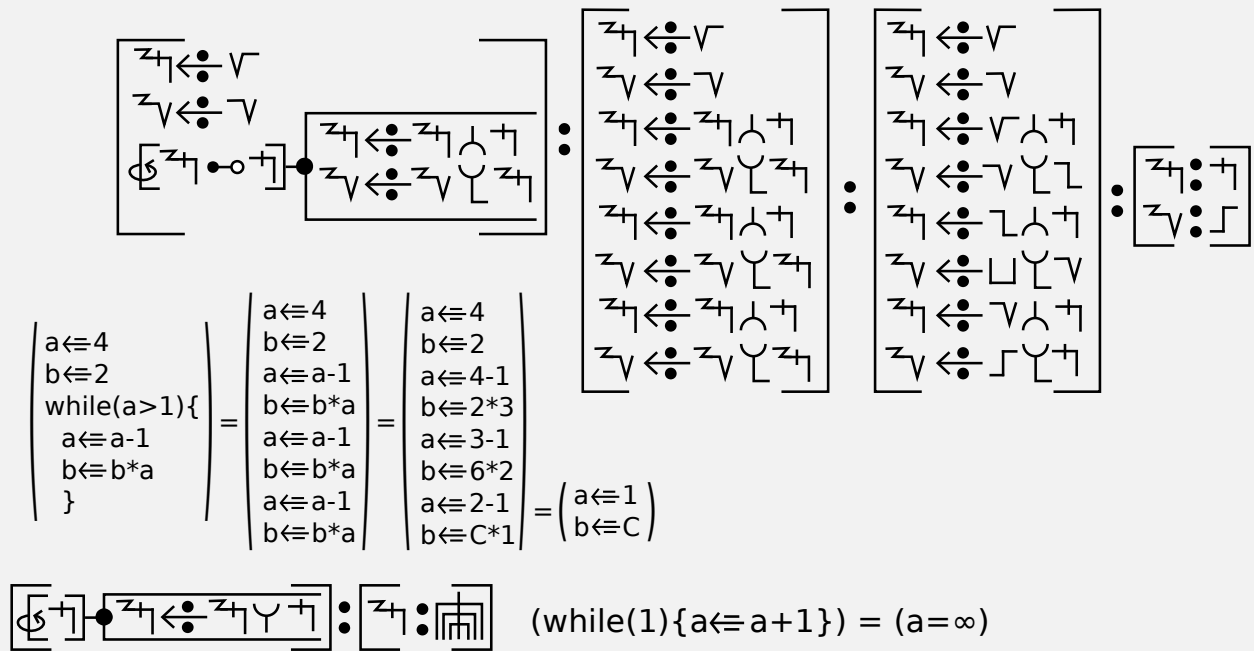
$$\left(\begin{array}{l} a <=> 0 \\ b <=> 1 \\ c <=> 2 \\ e <=> 3 \\ f <=> 5 \\ f <=> B \end{array} \right) = \left(\begin{array}{l} a <=> 0 \\ b <=> 1 \\ c <=> 2 \\ e <=> 3 \\ f <=> B \end{array} \right)$$

Array element reference

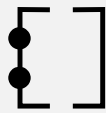


This establishes a format for assign and reference array elements individually. The "array element" ([] square bracket) is a closed box "sub" symbol

Loops



Execution & Return value

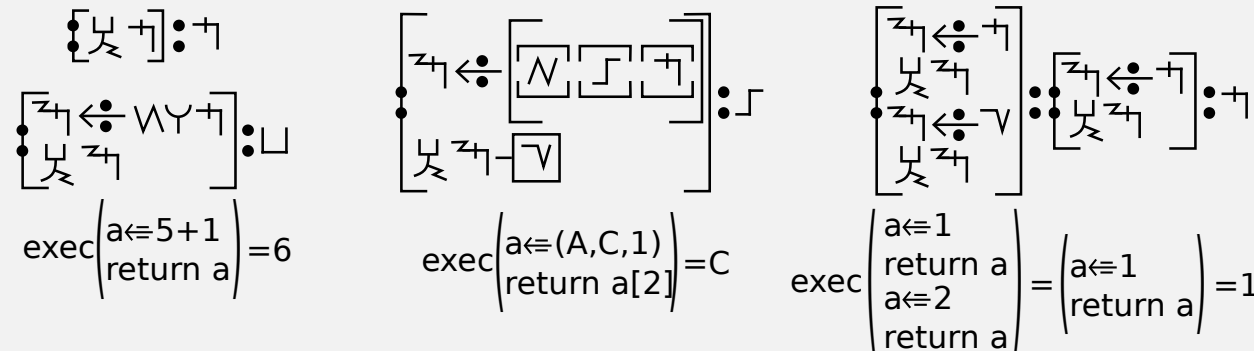


Execution Brackets



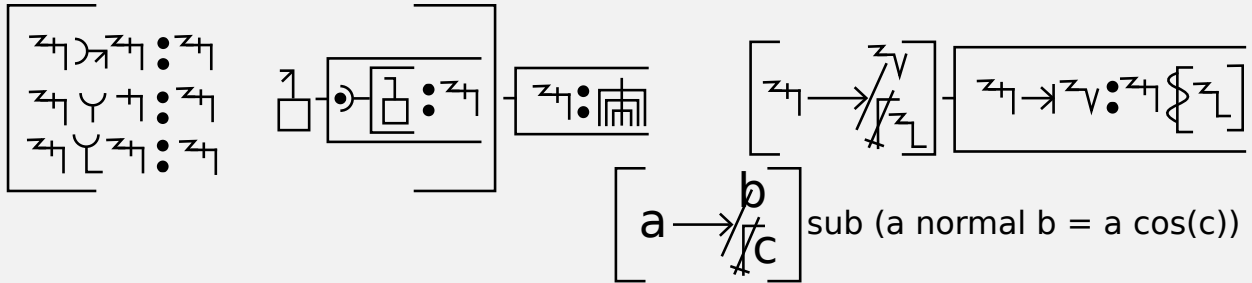
Return symbol

exec(return 1) = 1

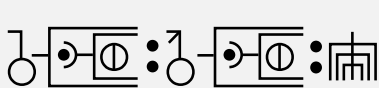


Infinity / Subspace / Normal vector / Perfect roundness

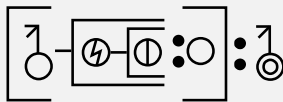
$$\left(\begin{array}{l} a^{\wedge}a=a \\ a+a=a \quad \text{3D cuboid sub(count(2D rectangle) = a)} \\ a*a=a \end{array} \right) \text{sub}(a=\infty)$$



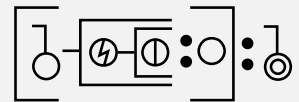
(3D space sub($\sigma(\text{diameter})=0$)
= perfect sphere



2D space sub(count(diameter))
= 3D space sub(count(diameter))
= ∞



(2D space sub($\sigma(\text{diameter})=0$)
= perfect circle

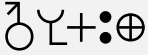




Infinity has already been defined with loops, here we clarify

The \updownarrow symbols will be used for general 2D/3D space


We will use \updownarrow for spheres and circles

4D sub(space-time)

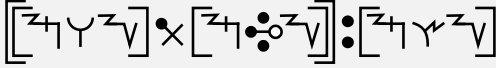
 3D space * time = space-time Charge = Charge

Here we establish that Space-time is the 4D product of 3D space and time
 We also define the construction of the charge symbol. It is a combination of "interaction" and "EM field".



 ((a-b=c) AND (c>b))=(a emit/radiate b)

Radiate/Emit and Absorb are defined as special cases of subtract and add



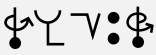
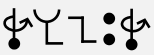
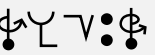
 ((a+b=c) AND (a>b))=(a absorb b)

Big things absorb smaller things

Breaking apart, smaller things are radiated by larger things






"Radiate/emit" are "merge" and "fork" symbols except that the smaller side is a wave

Half-integer-spin *3 = Half-integer-spin

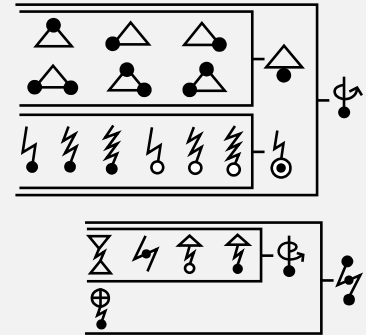
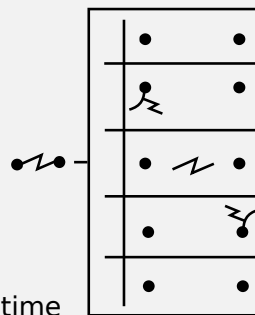




Here we establish a basic property of spin


Half-integer-spin *2 = Integer-spin Half-integer-spin *4 = Integer-spin


-  Gluon "a wave between chromodynamics"
-  Photon "wave-particle"
-  W-Boson "Hadron wave charge"
-  Z-Boson "Hadron wave no-charge"
-  Higgs "spacetime wave particle" gives mass which bends spacetime


example of emit and absorb




particle categories


 Boson "interaction-particle"


 Gauge Boson "integer spin particle"

 Lepton electron = "wave particle" more waves = more mass
 True-Dot = has charge False-Circle/0 = No charge

 Fermion "1/2 integer spin particle"




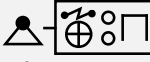
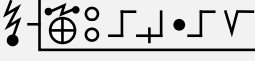

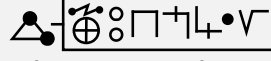
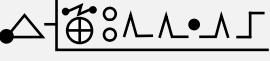
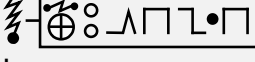
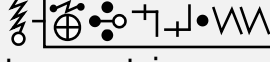
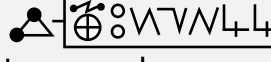
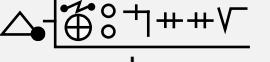



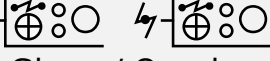
 electron
  muon
  tau
  electron neutrino
  muon neutrino
  tau neutrino

 Quark (triangle particle)
 Triangle represent 3 color charges of chromodynamics
 Dots on triangle represent 1/3 charge

 Up
  Down
  Charm
  Strange
  Top
  Bottom

Particles are defined via mass next, symbolism here is not necessary to decode symbols

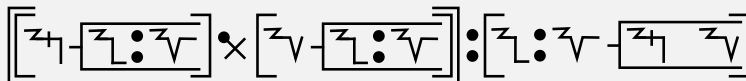
Particles & Mass

 electron sub(mass=1)	 electron neutrino sub(mass<484/10 ⁷)	 up quark sub(mass≈4.4E2)	 down quark sub(mass≈9)
 muon sub(mass≈CE.C4)	 muon neutrino sub(mass<553/10 ³)	 charm quark sub(mass≈917.4)	 strange quark sub(mass≈BB.DE)
 tau sub(mass≈CE.C4)	 tau neutrino sub(mass<1E.55)	 top quark sub(mass≈52A77)	 top quark sub(mass≈1FF4)
 Z-Boson sub(mass≈2B916)	 W-Boson sub(mass≈26673)	 Higgs Boson sub(mass≈3BB8A)	 Gluon / Quark sub(mass≈0)

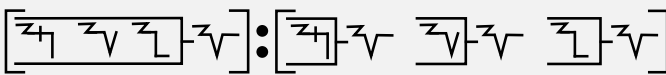
All units are arbitrary
 But the ratios between these values are universal, no matter what units used

The symbol for Mass is defined here via the value ratios same as the particles
 The symbolism for the mass symbol "interact with space-time"

Charge & "and" / "or" structures



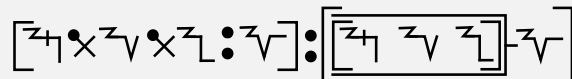
((a sub(c=d))AND(b sub(c=d)))
 =(a b)sub_of c=d



((a b c)sub_of d)
 =((a)sub_of d (b)sub_of d (c)sub_of d)



(a XOR b XOR c=d) = ((a b c)sub_of d)



(a AND b AND c=d) = ((a b c) sub_of d)

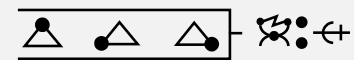
On the left we define some ways of using logical gates to represent conversational versions of "and"/"or"



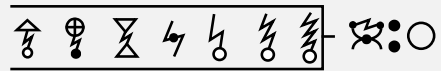
(W-Boson Muon Electron Tau) sub_of charge=1



(up charm top) sub_of charge=-2/3



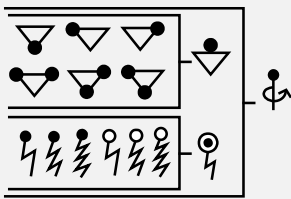
(down strange bottom) sub_of charge=1/3



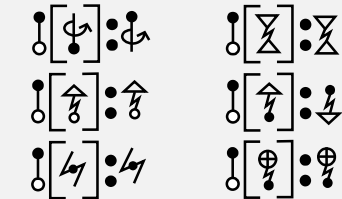
(W-Boson Higgs Gluon Photon Electron neutrino Muon Neutrino Tau neutrino) sub_of charge=0

On the right we define particle charge values. This defines the symbol for charge

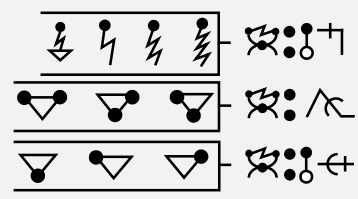
Anti particles



Antiparticle categories

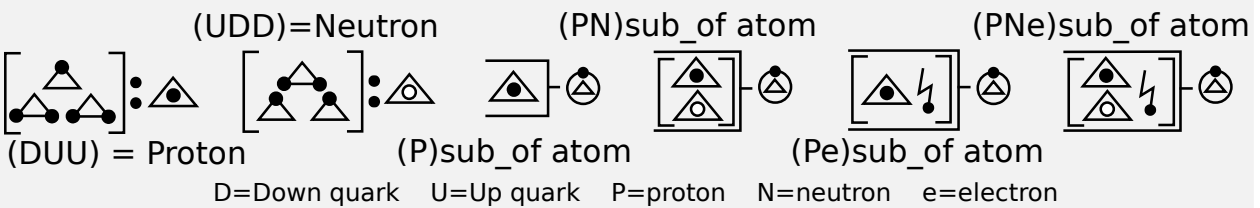


invert(fermion) = anti-fermion
 invert(gluon) = gluon
 invert(Z-boson) = Z-boson
 invert(W-boson-) = W-boson+
 invert(photon)=photon
 invert(Higgs)=Higgs

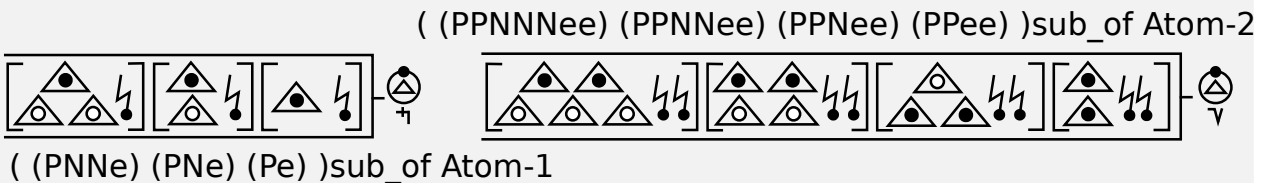


(W-boson+
 electron
 muon
 tau) sub_of charge=-1
 (up charm top)sub_of charge=2/3
 (down strange bottom)sub_of charge=-1/3

Hadron & Atoms

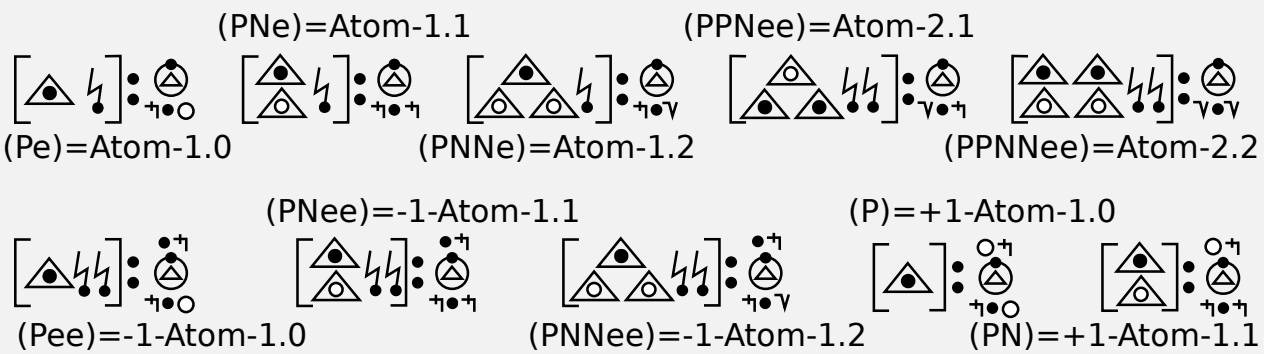


D=Down quark U=Up quark P=proton N=neutron e=electron

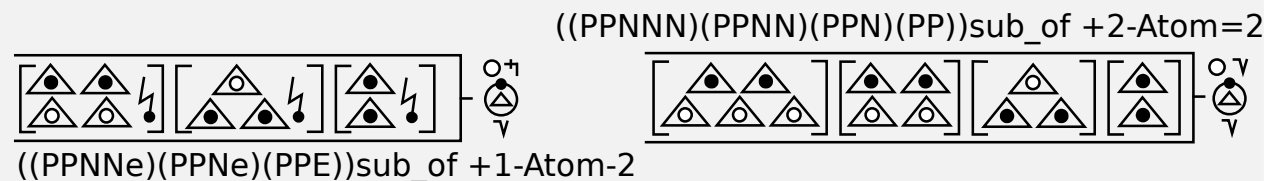
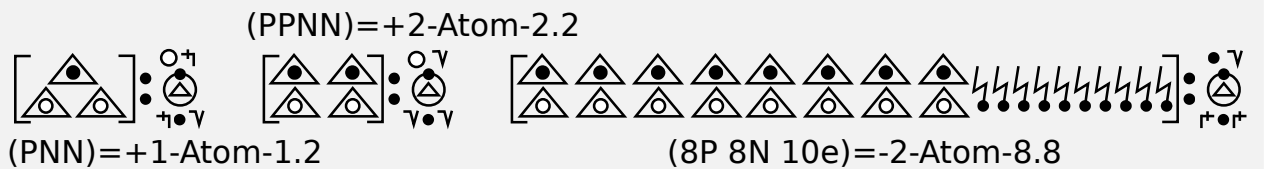


Atom-1 = Hydrogen, Atom-2=Helium, Atom3=Lithium, etc...

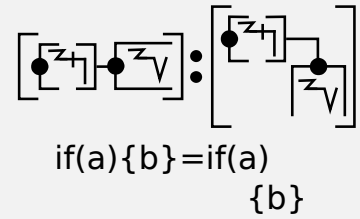
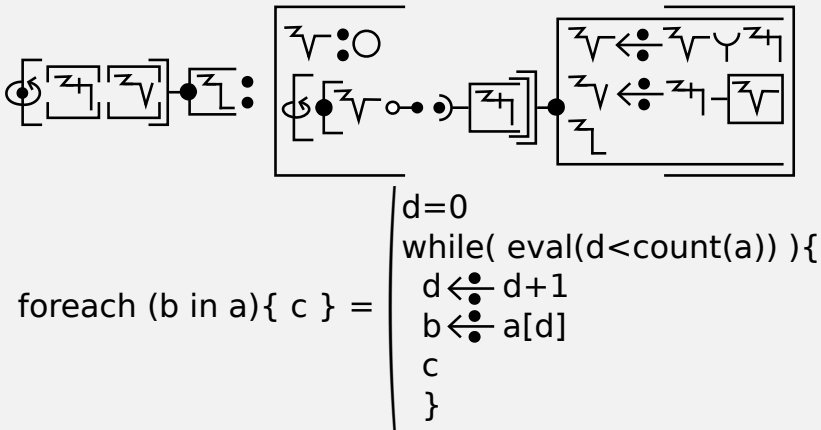
Isotopes and Ions



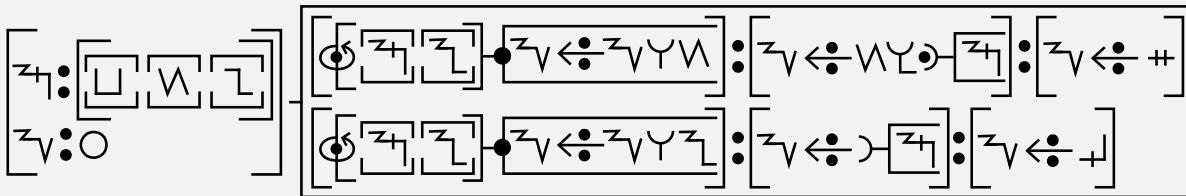
The dot on top denotes extra electron, the circle denotes missing electrons
 In the text labels we mark them with charge value



Foreach loops



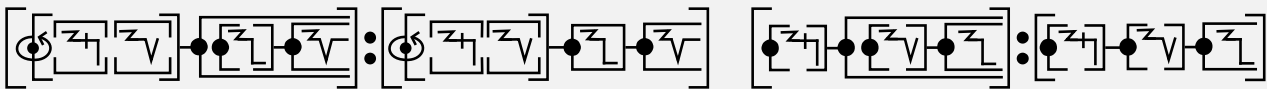
Just an example to show that expression connection lines are flexible in 2D



$$\left(\begin{array}{l} a=(6,5,3) \\ b=0 \end{array} \right) \text{sub} \left(\begin{array}{l} (\text{foreach}(a \text{ as } c) \{ b \leftarrow b+5 \}) = (b \leftarrow 5 * \text{count}(a)) = (b \leftarrow F) \\ (\text{foreach}(a \text{ as } c) \{ b \leftarrow b+c \}) = (b \leftarrow \text{sum}(a)) = (b \leftarrow E) \end{array} \right)$$

This gives us a clearly defined structure for using "foreach" operations

Nesting



$$\begin{aligned} & (\text{foreach}(b \text{ in } a) \{ \text{if}(a) \{ b \} \}) \\ & = (\text{foreach}(b \text{ in } a) \text{if}(a) b ;) \end{aligned}$$

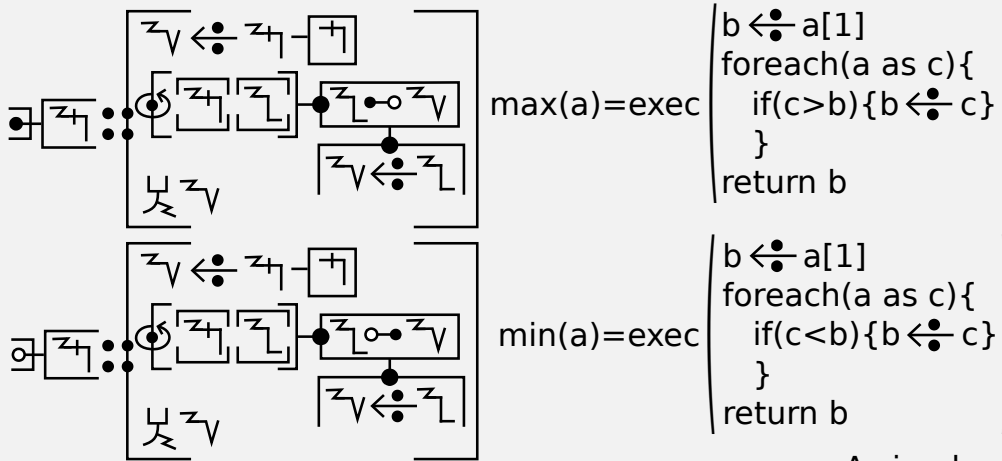
$$\begin{aligned} & (\text{if}(a) \{ \text{if}(b) \{ c \} \}) \\ & = (\text{if}(a) \text{if}(b) c ;) \end{aligned}$$



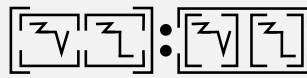
$$(\text{if}(a) \text{if}(b) c ;) = (\text{if}(a) \text{if}(b) c ;)$$

Here we demonstrate a few simple ways to simplify nested expressions. We also clarify that the eval brackets can be closed into boxes, this will help nested expressions and complex algorithms be drawn like flow-charts

Min / Max



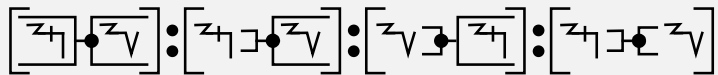
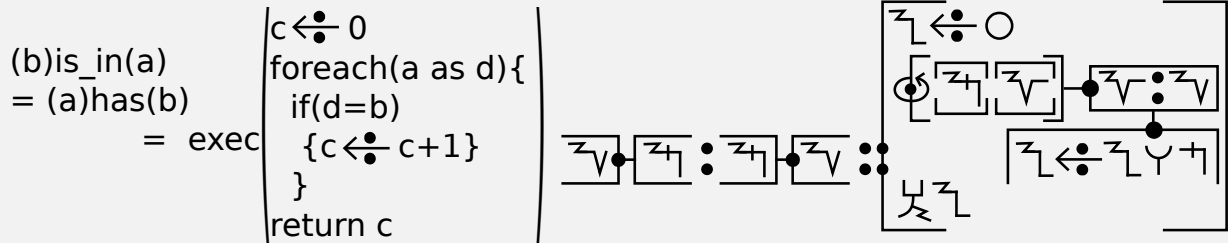
A simple example to show array element brackets just need to be rotated relative to the parent bracket



This defines the min / max functions

The simply return the minimum value or maximum value in an array

Has / Contains

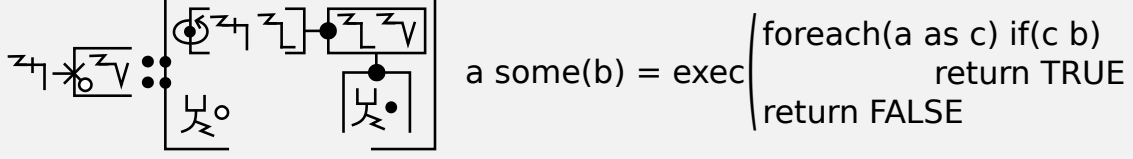
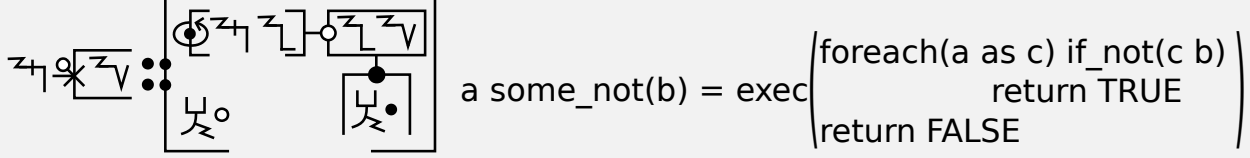
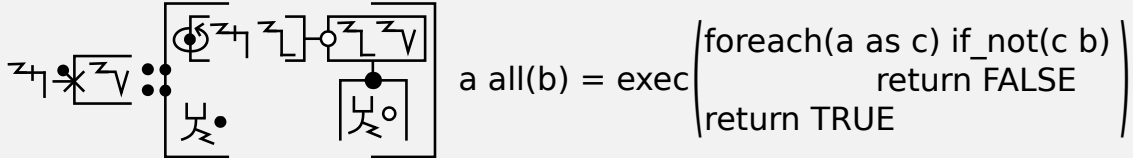
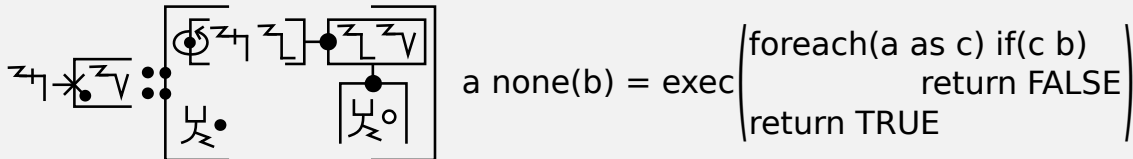


((a)has(b)) = (a has(b)) = (b is_in(a)) = (a has b)

Here we define a structure that can be used for "has/contains" and "is in"
It scans "a" for "b"

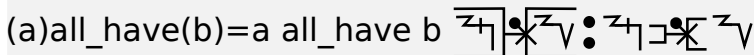
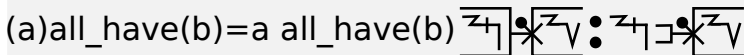
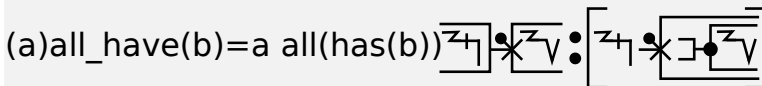
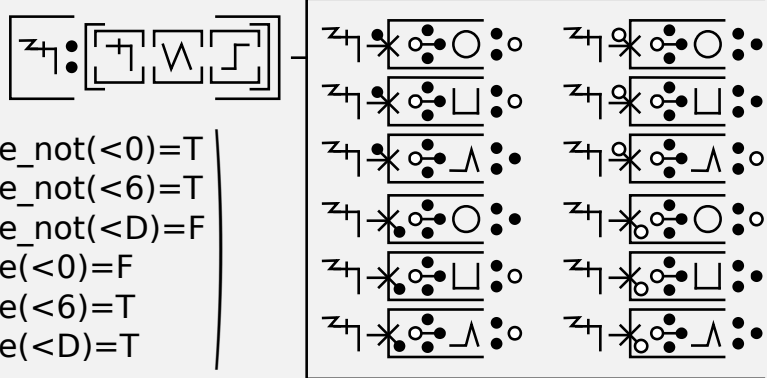
it returns the count of how many times it found "b" in "a"

All / None / Some / Some-Not



(a={1,5,C})

sub	a all(<0)=F	a some_not(<0)=T
	a all(<6)=F	a some_not(<6)=T
	a all(<D)=T	a some_not(<D)=F
	a none(<0)=T	a some(<0)=F
	a none(<6)=F	a some(<6)=T
	a none(<D)=F	a some(<D)=T



Here we define function for evaluating arraying.

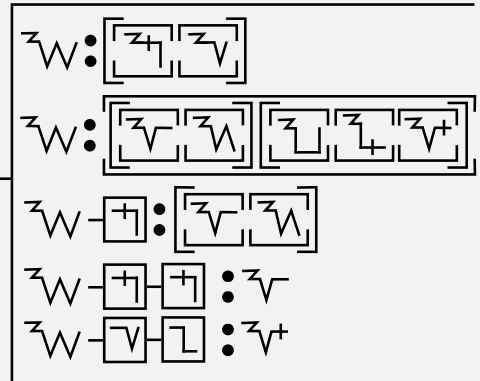
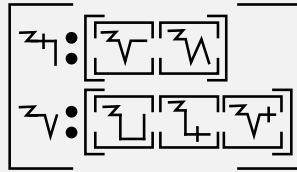
all, none, some and some_not take perform and expression on each element

all_have, none_have, some_have, and some_not_have check each check a 2D array to see if each top level element contain something

Multi-dimensional arrays

$$\begin{aligned} a &= \{d, e\} \\ b &= \{f, g, h\} \end{aligned}$$

$$\text{sub} \begin{cases} j = \{a, b\} \\ j = \{ \{d, e\}, \{f, g, h\} \} \\ j[1] = \{d, e\} \\ j[1][1] = d \\ j[2][3] = h \end{cases}$$

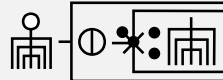
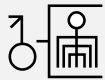


This example defines the basics of multi-dimensional arrays. It is a simple extension of single dimensional array structures.

Array handling to language abstraction

universe

sub(diameters all_T(=infinity))



$$\left[\begin{array}{c} z_1 \\ z_2 \end{array} \right] * \left[\begin{array}{c} z_3 \\ z_4 \end{array} \right] : \left[\begin{array}{c} z_1 \\ z_2 \end{array} \right] * \left[\begin{array}{c} z_3 \\ z_4 \end{array} \right] : \dots = ((a) \text{all_has_T}(b)) = (a \text{all_has}(b)=T)$$

3D space

sub(universe)

$$\left[\begin{array}{c} z_1 \\ z_2 \end{array} \right] * \left[\begin{array}{c} z_3 \\ z_4 \end{array} \right] : \left[\begin{array}{c} z_1 \\ z_2 \end{array} \right] * \left[\begin{array}{c} z_3 \\ z_4 \end{array} \right] : \dots = (a \text{all_T}(b)) = (a \text{all}(b)=T)$$

$$\left[\begin{array}{c} z_1 \\ z_2 \end{array} \right] * \left[\begin{array}{c} z_3 \\ z_4 \end{array} \right] : \left[\begin{array}{c} z_1 \\ z_2 \end{array} \right] * \left[\begin{array}{c} z_3 \\ z_4 \end{array} \right] : \dots = (a \text{some_not_T}(b)) = (a \text{some_not}(b)=T)$$

Here we create versions of array handlers that can express truths/facts

Next, on left, we define universe as "3D space whose diameters are infinite"

$$\left[\begin{array}{c} \text{atom} \\ \text{proton} \end{array} \right] * \left[\begin{array}{c} \text{atom} \\ \text{proton} \end{array} \right] : \left[\begin{array}{c} \text{universe} \\ \text{atom} \end{array} \right] * \left[\begin{array}{c} \text{atom} \\ \text{proton} \end{array} \right] : \dots = ((\text{atom}) \text{all_has_T}(\text{proton})) = ((\text{universe}[\text{atom}]) \text{all_has_T}(\text{proton}))$$

$$\begin{aligned} &(\text{atom sub}(\text{neutron all_T}(\text{bond proton}))) \\ &= (\text{universe sub}(\text{atom sub}(\text{neutron all_T}(\text{bond proton})))) \end{aligned}$$

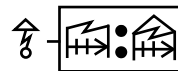
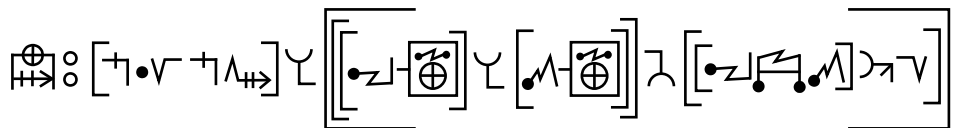
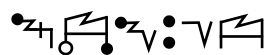
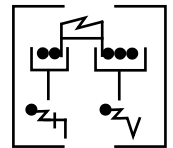
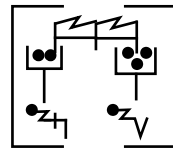
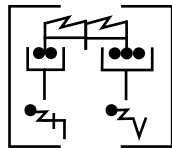
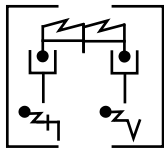
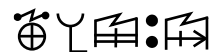
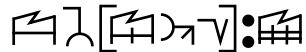
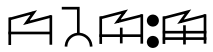
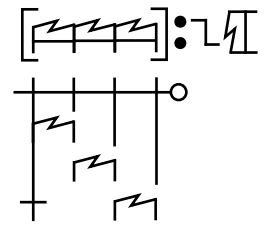
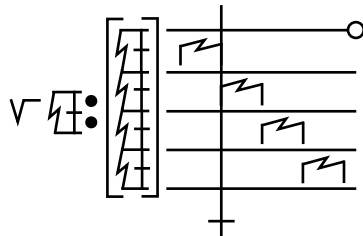
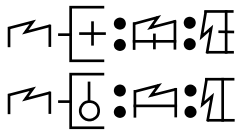
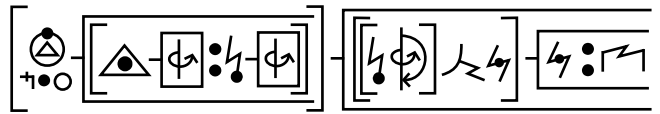
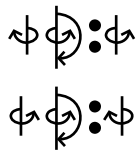
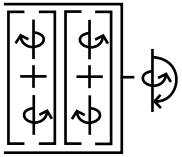
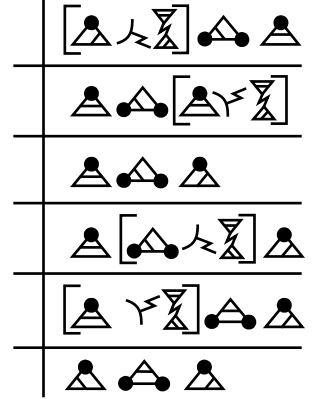
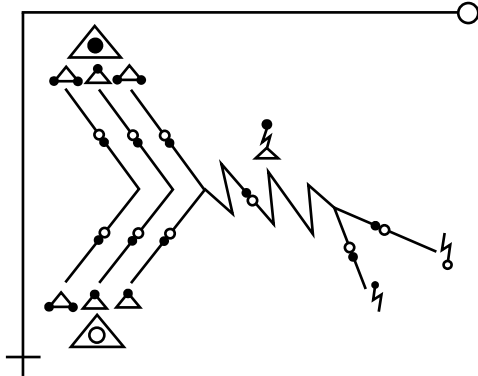
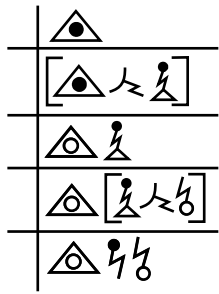
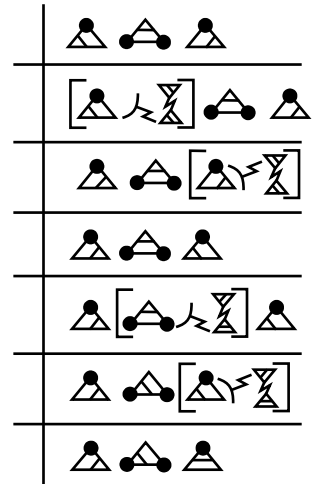
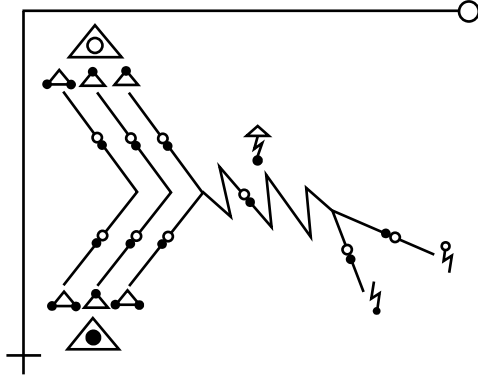
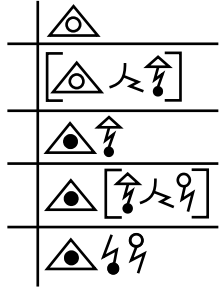


$$(\text{atom-1.0}) \text{none_has_T}(\text{neutron}) \quad \left[\begin{array}{c} \text{atom} \\ \text{neutron} \end{array} \right] * \left[\begin{array}{c} \text{atom} \\ \text{neutron} \end{array} \right]$$

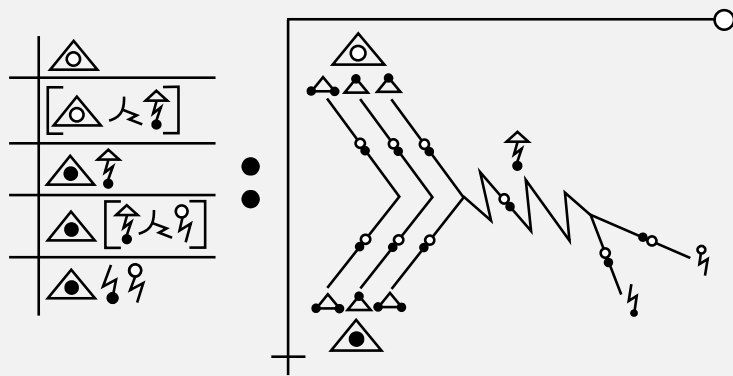
$$(\text{atom-1.1}) \text{all_has_T}(\text{neutron}) \quad \left[\begin{array}{c} \text{atom} \\ \text{neutron} \end{array} \right] * \left[\begin{array}{c} \text{atom} \\ \text{neutron} \end{array} \right]$$

$$(\text{atom}) \text{some_has_T}(\text{neutron}) \quad \left[\begin{array}{c} \text{atom} \\ \text{neutron} \end{array} \right] * \left[\begin{array}{c} \text{atom} \\ \text{neutron} \end{array} \right]$$

These examples demonstrate that we can use array handlers in descriptions of real world things. We also define that referring to a type of thing instead of a specific one means you are referring to "all of X in the universe"



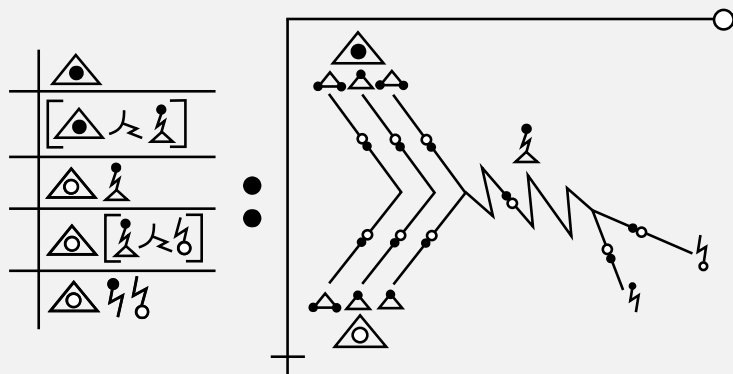
Feynman Diagrams, Weak & Strong



Here are Feynman diagrams Of the weak interaction.

The top depicts neutron decay via a W- boson
neutron decay

The bottom depicts a Proton turning into a neutron via W+ boson
electron capture



The vertical axis is labeled with a time cross, and the horizontal axis is labeled with a space circle.

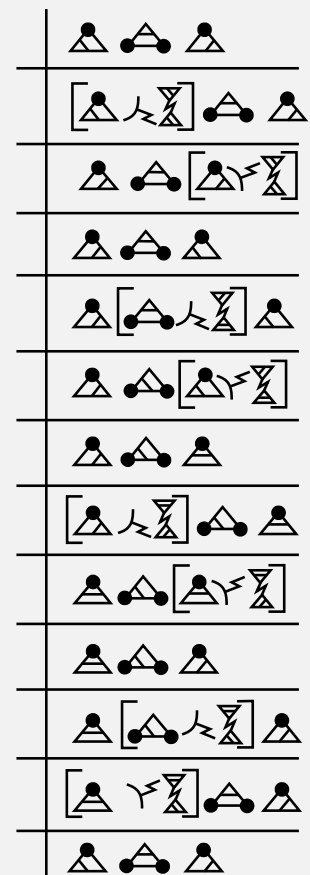
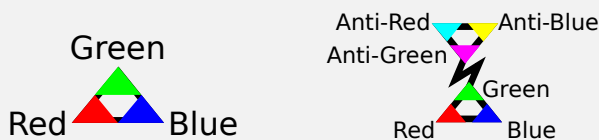
There work the same as standard Feynman diagrams with just a few cosmetic adjustments to fit Uscript

On the right you see a timeline which shows the Strong interaction that binds a neutron

The details of chromodynamics are complex, this is a simplified interpretation.

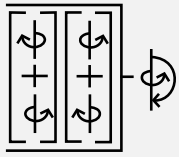
This example shows how to mark color charge on quars and gluons. each corner represents a charge.

Corners are divided into top / left / right.



These examples use fundamental physical processes to help define how we label graphs, describe process steps, and usage of radiate/emit and absorb symbols.

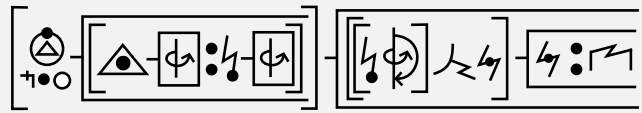
Units of Time and Distance



Spin Flip
(2 Examples of spin switching to opposite direction)



Spin flip usage examples

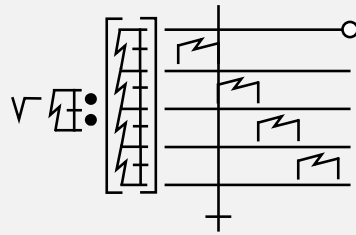


(Neutral Hydrogen sub(proton[spin]=electron[spin]))
sub (((electron spin-flip) emit photon)
sub(photon=unit photon)))

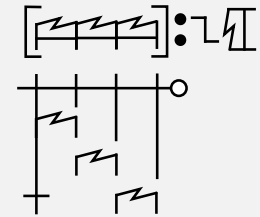


unit-photon sub(time)
= time unit = time unit

unit-photon sub(1D space)
= distance unit = distance unit



Time/Space graph example of time-unit

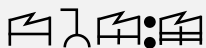


Time/Space graph example of space-unit

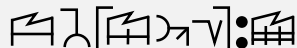
Here we define spin flip, hydrogen line photons, and how we use them as our units of space and time.

This is the measure units used on the voyager and pioneer plaques.

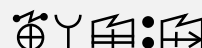
Speed, Acceleration, Momentum & Force



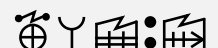
distance / time
= speed



distance / (time²)
= acceleration



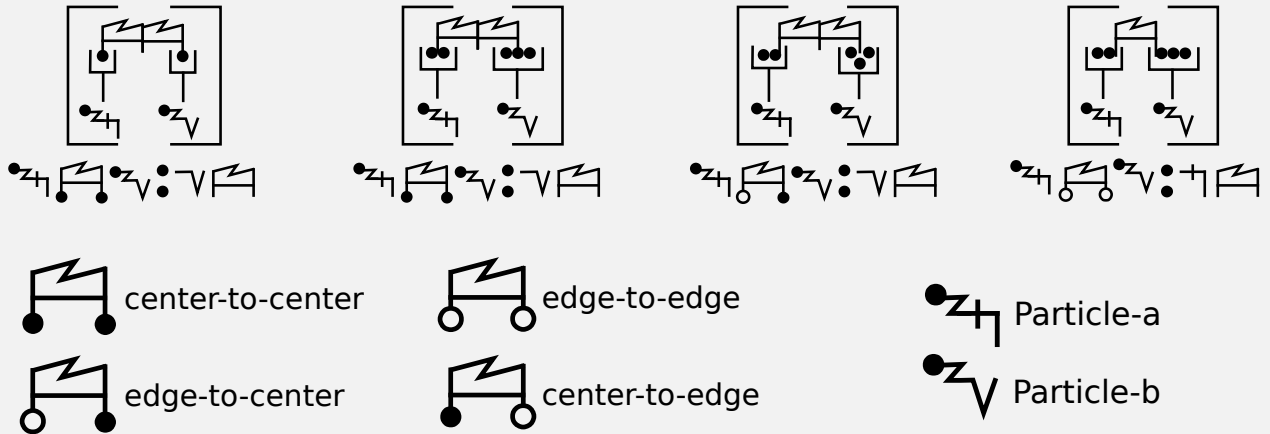
mass / speed
= momentum



mass / accel
= force

Now that we have distance, time, and mass units we can define some of the basic derived units

Distance Measurement & Particle variables

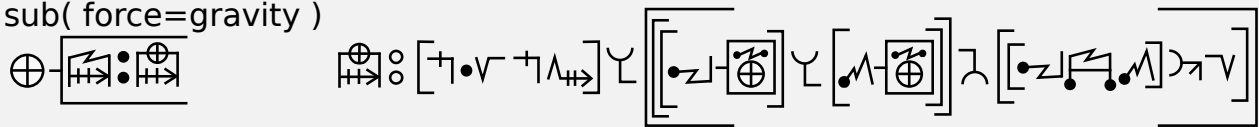


Here we define symbols to take measurements and "particle variable symbols"
 Measurement can be done from center or edge to center or edge
 Particle variables symbols make particles visually distinct in expressions

Gravity

Space-time

sub(force=gravity)



$$\text{Gravity} \approx (1.41 \times 10^{-2} F) * \frac{((a\text{-mass}) * (b\text{-mass}))}{(a \text{ center-to-center } b)^2}$$

Here we define gravity using Newtons equation for gravity

This is accurate enough for most purposes

The gravitational constant has been converted for use with our units

6.67408×10^{-11} only works for meters / kilograms / seconds

Our units are

mass : 9.109×10^{-31} Kg (mass of 1 electron)

distance : 0.211061140542 meters (Hydrogen line wavelength)

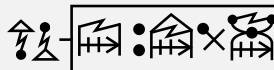
time : 0.7040241837 nanoseconds (Hydrogen line frequency)

Boson forces

Photon
sub(force=EM-force)



EM charge
sub(force=EM-force)



W+/- boson
sub(force =
Strong-force
AND
EM-force)

Z boson
sub(force = Weak-force)



Gluon
sub(force
=Strong-force)

Here we describe force carriers

No higgs because it "gives mass" it does not "mediate gravity"

△H⚡ : △H⚡ : ⊕

△K∇ : △H∇ : ☆

△-H-△ : △-H-△ : △

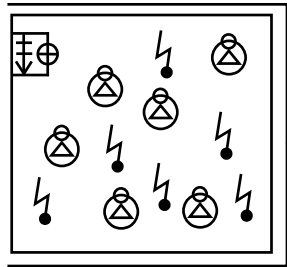
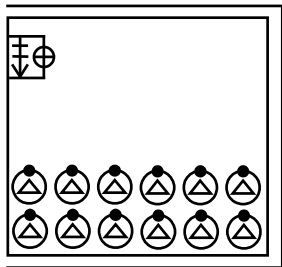
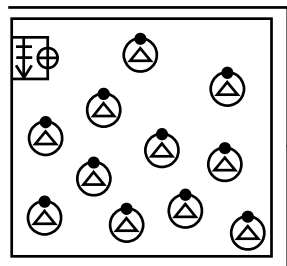
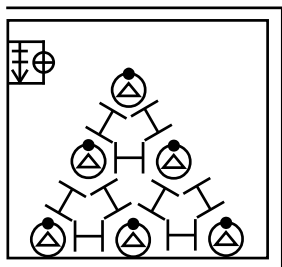
△-H-△ : △-H-△ : ⊕

△-H-△ : △-H-△ : ⊕

△-H-△ : △-H-△ : ⊕

△-H-△ : △-H-△ : ⊕

⊕ : ⊕ : ⊕



[⚡ : ⊕ : ⊕]

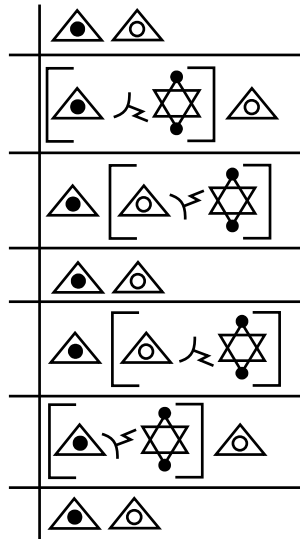
⊕ : ⊕ : ⊕

[⊕] : ☆

[⊕] : ☆

[⊕] : ☆

[⊕] : ☆

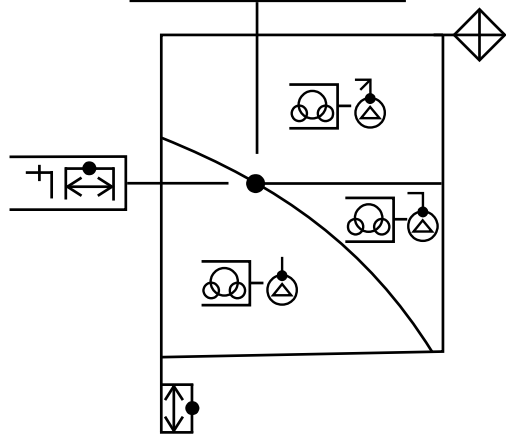


⊕ : ⊕

⊕ : ⊕ : ⊕

⊕ : ⊕ : ⊕

⊕ : ⊕ : ⊕

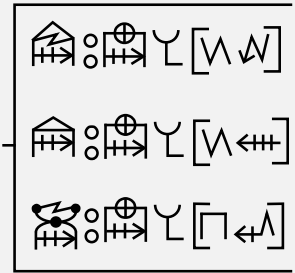
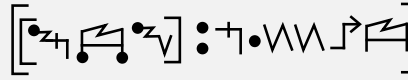


⊕ : ⊕ : ⊕

Relative strength of forces

$$\left(\frac{\text{Particle-a}}{\text{Particle-b}} \right)_{\text{center-to-center}} = 1.55/10^{12}$$

sub(
 Weak-force \approx gravity*(5*10^A)
 Strong-force \approx gravity*(5*10^1F)
 EM-force \approx gravity*(9*10^1D)
)



The above translates to :

“At a distance of 1 femtometer the weak force is approx 10^{32} stronger than gravity, the strong force is approx 10^{38} stronger than gravity, and the electromagnetic force is approx 10^{36} stronger than gravity.”

***Warning : We don't really know much about gravity at these scales!**

Newtons equation for gravity is only “accurate enough” within certain scales and ranges. So the accuracy of this statement about gravity at this scale is not known.

So either

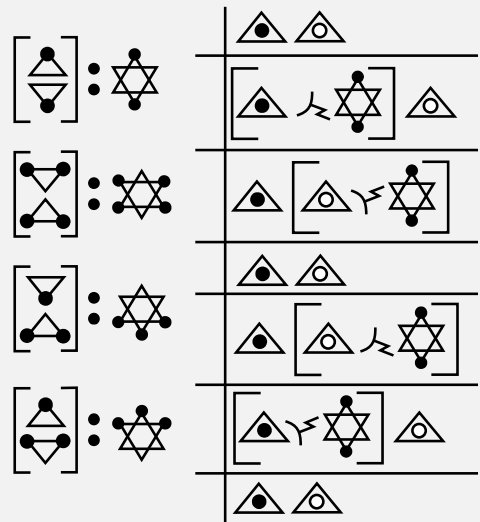
- a) accept that gravity here is defined by newtons equation and these ratios are based on that
- b) just don't use this table and instead define the forces independently
- c) skip it, if you are not discussing the forces in detail this is not really necessary

Mesons

On the right:

4 examples of how to combine quark symbols to produce meson symbols

A timeline of a Proton and Neutron exchanging mesons. This is a simple description of the mechanism binding atomic nuclei together



*It is technically not “correct” to think of the Up/anti-Up and Down/anti-Down as separate particles, we could define that now, but that's a much deeper conversation for another day.

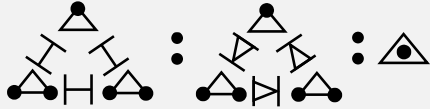
Bonds



proton bond electron = proton EM-bond electron = Atom-1.0



down strong-bond anti-down = down bond anti-down = down-antidown-meson



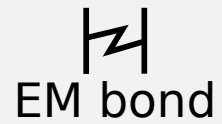
up up down (all bond to each) = up up down (all strong-bond to each) = proton



N bond P bond e = N strong-bond P EM-bond e = Atom-1.1

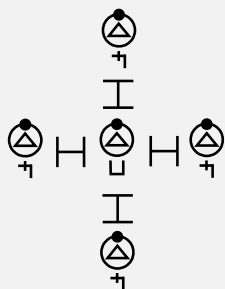


NPPe (NPP bonded, Ps bonded to e)
 =NPPe (NPP strong-bonded, Ps EM-bonded to e)
 =-1-Atom-2.1

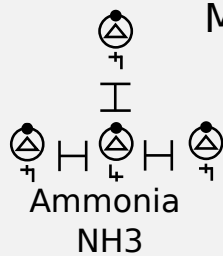


Here we define and establish symbols for bonds
 One general bond symbol, and specific ones for EM and strong force bonds

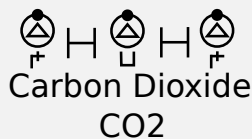
Molecules



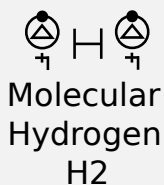
Methane
CH4



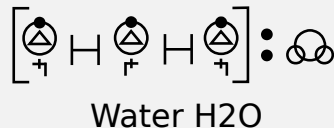
Ammonia
NH3



Carbon Dioxide
CO2



Molecular Hydrogen
H2

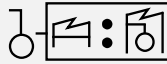


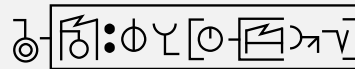
Water H2O

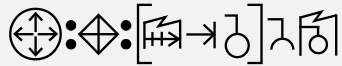
On the left we give a few examples of atoms bonded to form molecules

We also establish a symbol for water

Pressure


2D space sub (length unit = Area unit) 

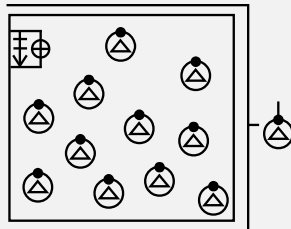
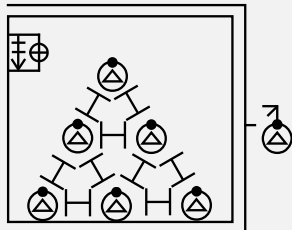
Circle sub(area= $\pi \cdot (\text{radius sub}(\text{length unit})^2)$) 


pressure=pressure=(force normal 2D space)*area 


Here we establish a symbols for area and pressure, and define pressure


States of Matter

 (H+ He+ He2+) sub_of cation

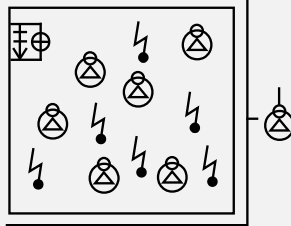
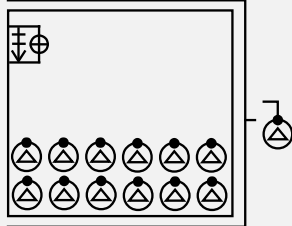


 **Solid**
3D-atom
atoms hold 3D structure

 **Liquid**
2D-atom
2D Surface tension

 **Gas**
1D-atom
Free Particles

 **Plasma**
1D-cation
Free Particles



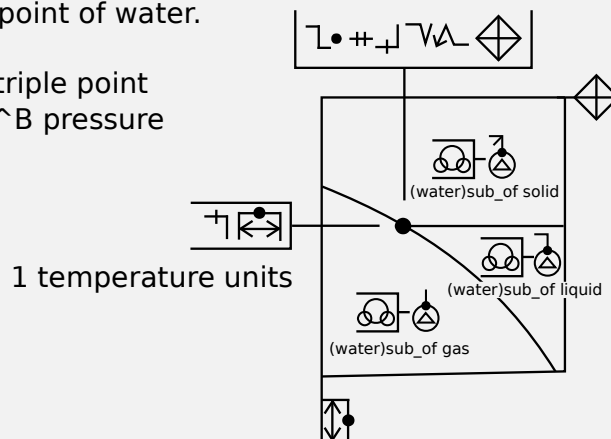
Gravity is marked by the force arrow in the gravity symbol inside each graphic

Temperature


Temperature, the final and hardest to define base unit, is defined by the triple point of water.

($3.15 \cdot 10^8$ Pressure units)

The graph to the right shows the triple point of water. it is labeled at $3.15 \cdot 10^8$ pressure units and 1 temperature unit.

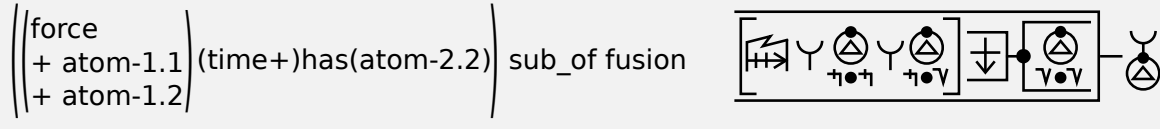
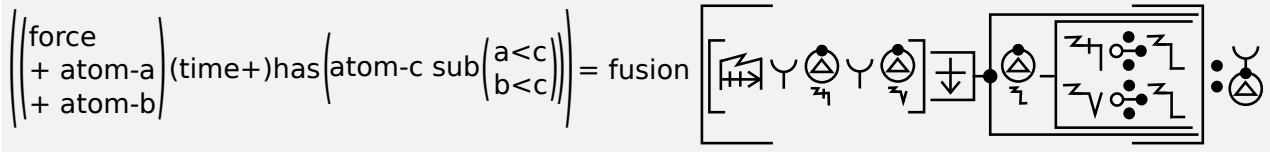


Below we define 0 temperature as absolute zero (atom speed=0). This gives us a well defined temperature scale


0 temperature units
= (atom sub(speed=0))

Fusion

$$\left[\left[\left[z_1 \downarrow z_2 \right] - \left[z_3 \cdot z_4 \right] \right] : \left[z_1 \downarrow z_2 \right] \right] \left((a \text{ time+ } b) \text{sub}((b) \text{has}(c)) \right) = (a \text{ (time+)has}(c))$$



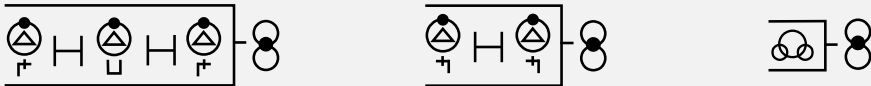
Here we have defined fusion as 2 atoms +force + time whose result contains an atom that is of a larger atomic number than any of the original atoms

- Since our atomic number can be read as a decimal number, that means:
- more protons is a larger number regardless of how many neutrons
 - if the protons are the same, more neutrons is still a larger number but
 - if protons are less then it is not fusion matter how many neutrons

This is perhaps not ideal, we may want to redefine it to use the atoms mass instead of its atomic number. For now I think this is good though since an atom with less protons and significantly more neutrons will not be a stable isotope.

Molecules

(H bond H)sub_of molecule




(O bond C bond O)sub_of molecule

(water)sub_of molecule

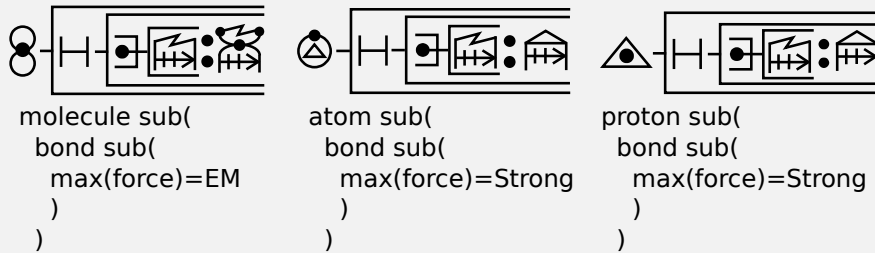
This defines that molecules are bonded collections of atoms

Systems and Bonds

system sub(molecule atom proton) 

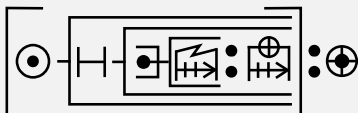
Here we establish a starting definition for "system" term we will use a lot. In Uscript "system" has a very broad scope.

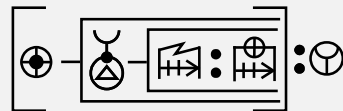
This has not defined "system" as generally and abstractly as we want, but it is enough for now. The more general meaning will be defined through more usage as we go forward.



On the left we show how to reference the internal bonds of a system. Here we also define the strongest binding forces of various systems

Astronomical bodies & Stars

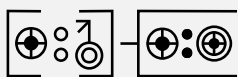
(system sub(bond sub(max(force) = gravity)))
= astro_body 


(astro_body sub(fusion sub(force=gravity))) = Star 

We define a symbol for an "astronomical body" as any system for which the strongest binding force is gravity..

This definition may exclude some small asteroids for whom the Van der Waals force can be a stronger binding force than gravity. I do not believe this is a problem, at some point between a moon and a clump of dust we must draw an arbitrary line, "gravity being the strongest binding force" works fine.

Next we define a "star" as any astronomical body in which atomic fusion is driven by gravity.

 (astro_body approx perfect sphere)
sub(astro_body = spherical astro_body)

 (eval_not(astro_body approx perfect sphere))
sub(astro_body = spherical astro_body)

Astronomical body orbits

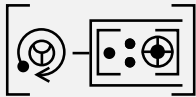
The first major classification is approximately spherical / non-spherical



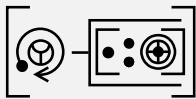
Star-orbiter

This symbol shows a particle orbiting a star using our existing defined symbols for "orbit/rotation/spin", "particle" and "star".

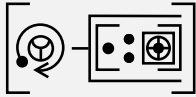
Orbiting a star is a very important class because bodies orbiting a star are gravitationally bound in orbits, have a consistent source of radiation and energy, and many other significant properties.



(star-orbiter sub(particle=astro_body))
sub(particle = star_orbiting-astro-body)



(star-orbiter sub(particle=spherical-astro_body))
sub(particle = spherical-star_orbiting-astro-body)



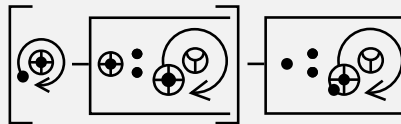
(star-orbiter sub(particle=non-spherical-astro_body))
sub(particle = non-spherical-star_orbiting-astro-body)



Astro-body-orbiter

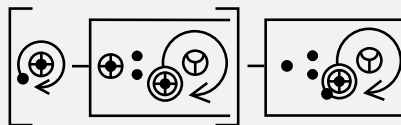
A body that orbits an astronomical body.

It's mildly implied that we mean "orbiting non-star bodies", and that is not necessary, but it can easily be added to the definition.

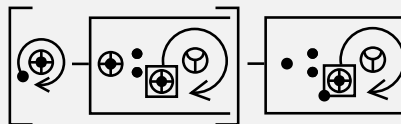


(astro-body-orbiter sub(astro-body = star-orbiting-astro-body))
sub(particle = star-orbiting-astro-body-orbiter

"particle" refers to the only particle in the statement, the one in the first symbol

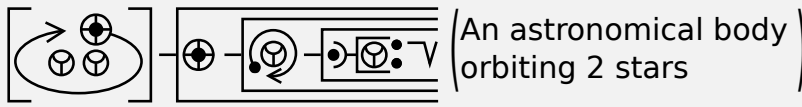


(astro-body-orbiter sub(astro-body = star-orbiting-spherical-astro-body))
sub(particle = star-orbiting-spherical-astro-body-orbiter



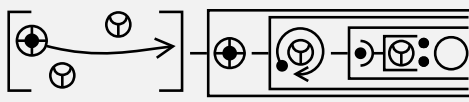
(astro-body-orbiter sub(astro-body = star-orbiting-non-spherical-astro-body))
sub(particle = star-orbiting-non-spherical-astro-body-orbiter

Host stars



(An astronomical body orbiting 2 stars)

```
sub(
  astro-body sub(
    star-orbiter sub(
      count(stars)=2
    )
  )
)
```



(An astronomical body passing by but not orbiting stars)

```
sub(
  astro-body sub(
    star-orbiter sub(
      count(stars)=0
    )
  )
)
```

This shows how to count the number of host stars of an orbiter
 We will use this to define bodies without a host star (eg. rogue planets)

```
astro-body sub( star-orbiter sub( count(stars=0) ) )
```

= free-astro-body

```
spherical-astro-body sub( star-orbiter sub( count(stars=0) ) )
```

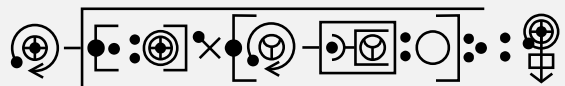
= free-spherical-astro-body

```
non-spherical-astro-body sub( star-orbiter sub( count(stars=0) ) )
```

= free-non-spherical-astro-body

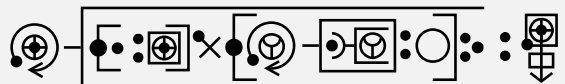
```
astro-body-orbiter
sub(
  astro-body-orbiter sub(
    eval(orbiter=spherical-astro-body)
    AND
    eval(
      star-orbiter sub(count(stars)=0)
    )
  )
)
```

=free-spherical-astro-body-orbiter




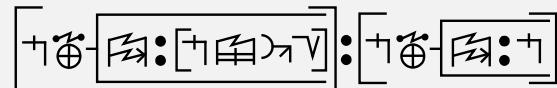
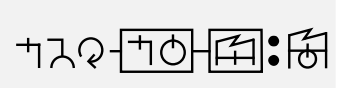
```
astro-body-orbiter
sub(
  astro-body-orbiter sub(
    eval(orbiter=non-spherical-astro-body)
    AND
    eval(
      star-orbiter sub(count(stars)=0)
    )
  )
)
```

=free-non-spherical-astro-body-orbiter

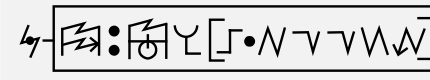


These give us a set of general categories of for astronomical bodies like planets, moons, asteroids, categorizing them based on what they orbit, and/or if they are approximately spherical.

Energy & Frequency

force * distance = energy
 1 mass sub(energy=1 speed ^ 2) = (1 mass sub(energy=1))
 1/cycle[1 revolution][time] = frequency




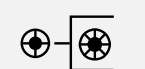
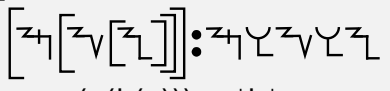
 photon sub(energy=frequency*C.A225*(10^A))

Here we define energy (force*distance).
 Next we define energy mass equivalence
 Then frequency, and finally clarify more
 by defining the energy of photons via their
 frequency

Uscript speed 1=speed of light
 so c=1... so c^2=1^2... so c^2=1
 multiplication by 1 does nothing
 so e=mc^2 is e=m

Galaxies & Black Holes

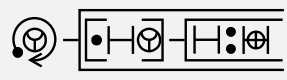
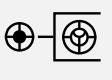
astro_body sub(black_hole)

 astro_body sub(neutron_star) (a(b(c)))=a*b*c

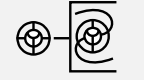
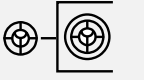
Here we introduce symbols for black hole and neutron star that will be defined in the next sections, and reinforce that no marked operation defaults to multiplication

astro_body sub(galaxy)

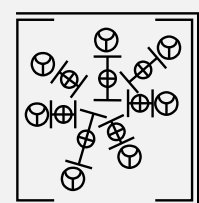
 star_orbiter sub(
 (orbiter bond star)sub(
 bond=gravity_bond
)
)
 (galaxy approx 3D)
 sub(galaxy=elliptical_galaxy)

galaxy sub(spiral_galaxy)

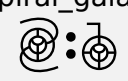

 galaxy sub(elliptical_galaxy)

(multiple stars gravitationally bound to common center region)




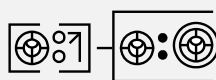
 sub_of galaxy

spiral_galaxy =spiral_galaxy

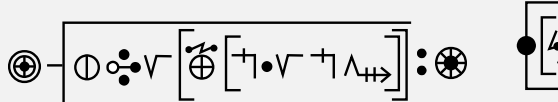



 elliptical_galaxy =elliptical_galaxy

(galaxy approx 2D)

 sub(galaxy=spiral_galaxy)

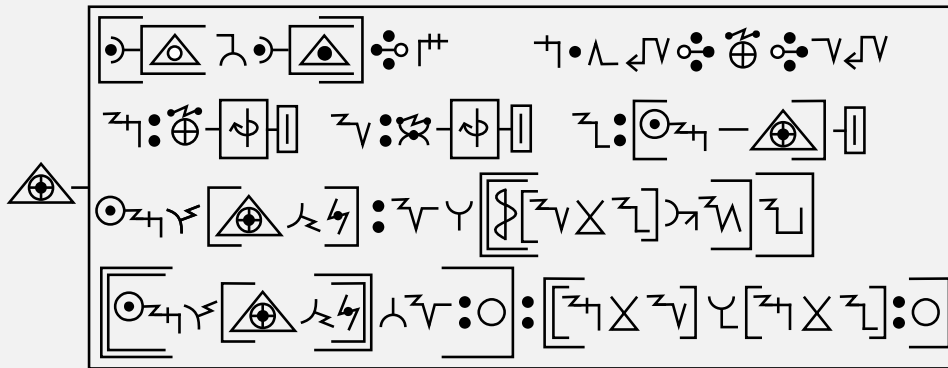


 spherical_astro_body
 sub(diameter<4(mass(1.41/(10^2F))))
 =black_hole

if(
 (photon distance black_hole)
 <
 black_hole sub(mass(1.41/(10^2F)))
)then{black_hole absorb photon}

This section defines galaxies as clusters of stars with a common gavitational center.
 2 categories of galaxy are defined, planar/spiral and elliptical(ellipsoidal)
 We also define black holes using the schwarzschild radius and event horizon

Neutron Stars



```

neutron_star sub(
  (count(neutron)/count (proton))>10   1.B*10^32 < mass < 2*10^32
  a=mass[spin][axis]  b=charge[spin][axis]  c=(system-a line neutron_star)[line]
  system-a absorb (neutron_star emit photon)=d+((cos(b angle c)^e)*f)
  ((system-a absorb(neutron_star emit photon))-d=0)=((a angle b)*(a angle c)=0)
)
  
```

This defines several qualities of neutron stars.

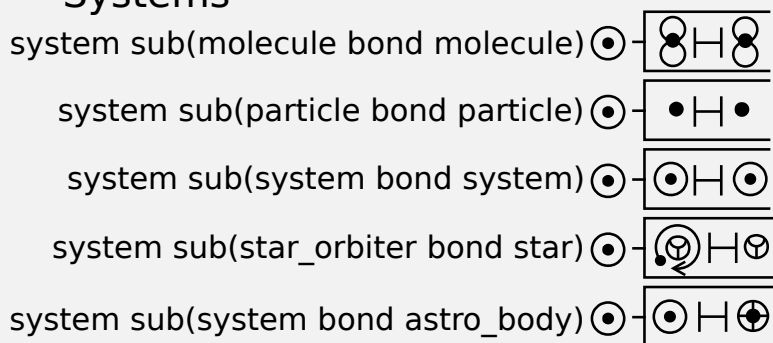
(line 1)1.ratio of protons to neutrons 2.mass range

(line 3)3.the observed brightness can pulse

(line 4)4.if it does not pulse then the magnetic axis is in line with rotational axis or the observer is in line with rotation axis. (either angle is 0)

Systems

Here we further clarify the broad meaning of the Uscript word "system" by include some more examples of larger systems.



This doc is being composed with a full Uscript key (pure self defined Uscript) one “page” at a time (determined by the dimension of standard document pages) The format will be as you see above, a page of the key followed by a breakdown and explanation.

Pages added in proper sequence, so at any point the Uscript is self-defined and can be decoded with the key pages alone. Each page only relies on previous pages to define itself.

Explanation pages can be removed and it still self-defines.

Uscript v1 at <http://www.dscript.org/>