Uscript

Table of contents

V2 – The Self-defining Key

This v1 doc provides detailed explanations and descriptions of the language, design, reasoning and the creative process.

Best to read this v1 doc first.

v2 is a self defining key with breakdown and short simple descriptions.

Check out v2 if to better understand how Uscript "universally self-defines"

Uscript v2: http://www.dscript.org/uscript2.pdf

v2 is fundamentally the same but there have been some basic revisions to numbers and math
v2 is still a work in progress, I am adding pages to it as I work on it.

Pre-read FAQ

Q: How is this a universal language?

A : Unlike most languages Uscript divorces itself form the human experience as much as possible. It starts by defining math visually in a universally obvious way, then physics, and then builds upon them to construct higher level and more abstract terms. Unlike other constructed languages it does not define itself with another language, its dictionary is entirely written in Uscript. This document does describe Uscript with English to help explain it to the reader but the English is not necessary The definitions are in the graphics, If you extract only the graphics Uscript can stand alone without any human language to define itself and be universally decipherable (as long as the reader has a minimum level of understanding in math and physics).

Q: What if other life forms have a different understanding of physics?

A: Math and Physics are Convergent. While language and culture are divergent, meaning if groups are separated they will tend to diverge and become different over time, Math and Physics are the opposite. There are levels or "resolutions" of understanding. For example, we believe an electron is a fundamental particle not composed of smaller bits, but we once thought this of atoms, regardless, I could still discuss atoms with someone who doesn't know about electrons and nuclei, and in the same way if another civilization can see that electrons are made of smaller things they will still understand our concept of electrons.

Q:What if physics is different in different places?

A:While environmental factors ranging from gravity fields to PH can affect interactions, here we use the fundamental laws of physics, there are many proofs that these are universal laws. To simplify : "If you change these laws then stars don't work! So anywhere we can see stars we know these laws are true, and we see stars at 13.3 billion light years away (furthest observed object)"

Q:Why do you start with such difficult subject matter?

A:Because I need to build some basic vocabulary and structure, the only way to do that is to start by defining universal laws and establish linguistic structure and vocab through examples of these laws. I can't start with pronouns because first I need concepts like communication, which needs concepts like interactions & models, which needs logic & programming which needs systems & formulas, etc. I can't just say " this symbol means 'you' and this symbol means 'me' ".

Q:So I need to understand all of this stuff to understand Uscript?

A:Not really. I cover many subjects in physics that are not used in any of the higher level concepts yet. You can easily skip ahead or just learn vocab, and it would roughly work, but without properly understanding the language & structure that is established in the Uscript definition layers it could translate poorly. Rather like Chinese characters, if you just translate characters one by one into English and string them into a sentence, your Chinese sentence is likely to be gibberish.

Q:Is this the full language

A:This is just the first few layers. It is built up to a level where some basic linguistic communications can be done. It takes TONS of time and work to build the vocab because you can't just start "making up words and typing in a definition in English", each term must be defined with graphics using only previous layers of Uscript. There is an insane amount yet undefined, I just touch the surface on few key topics, each topic could easily become a document of its own.

Q:Can I change/add symbols?

A:Of course, the symbol meaning is in the graphics. Feel free to add to it or modify it into your own version. I could spend a life time on it and barely scratch the surface of human knowledge. Uscript is under a Creative Commons license, free for any use, even commercial use, No royalty or Fee.

Q:But you use so much English to define you radicals, symbols etc.

A:Only in the first few pages because I am introducing the script and trying to fast track your understanding. "The English stuff is just for your benefit" not actually needed. For example, the graphic below defines the math section and the fundamental particles without any English at all.

      The numbers are visually obviously

      With numbers the equations allow you to easily infer the meaning of the operation symbols

      With operations and numbers the math constants and terms are defined

      With just numbers the particles are defined

         *The particles are defined not because those units are universal, but because the ratios of the values are the same no matter what units you use.
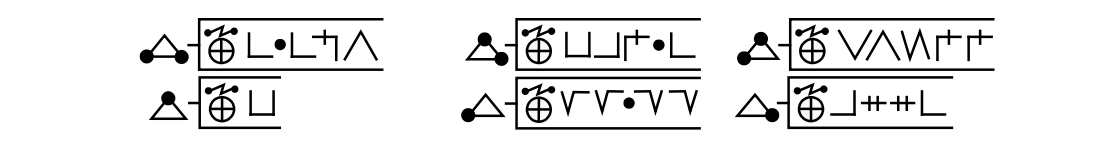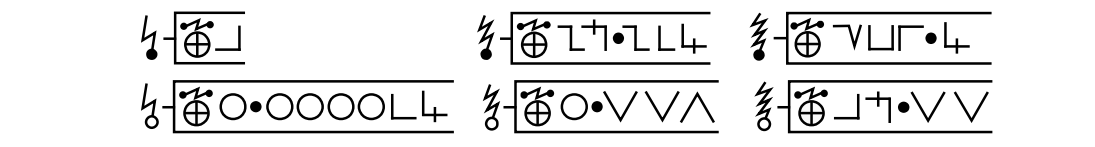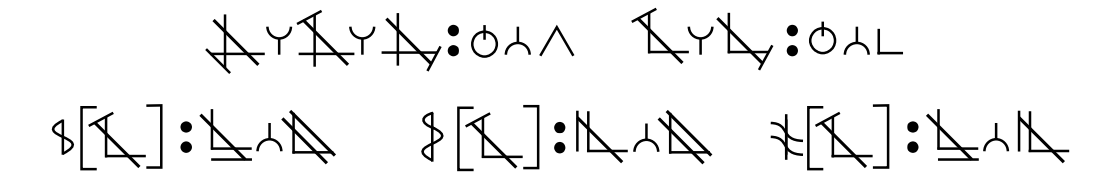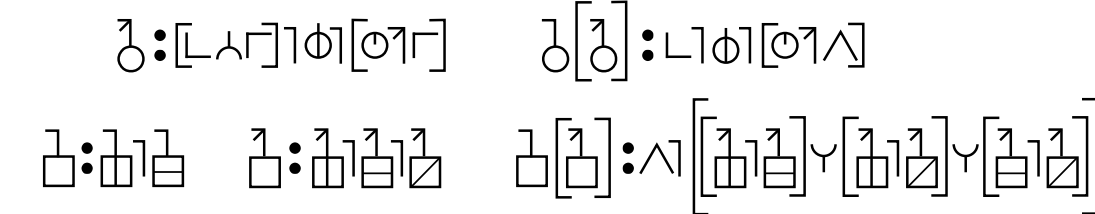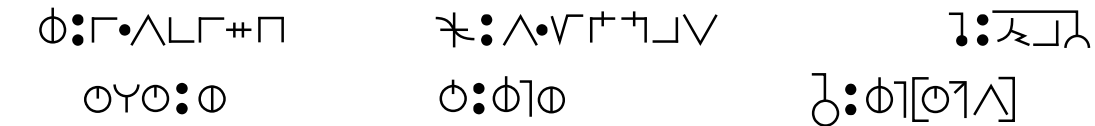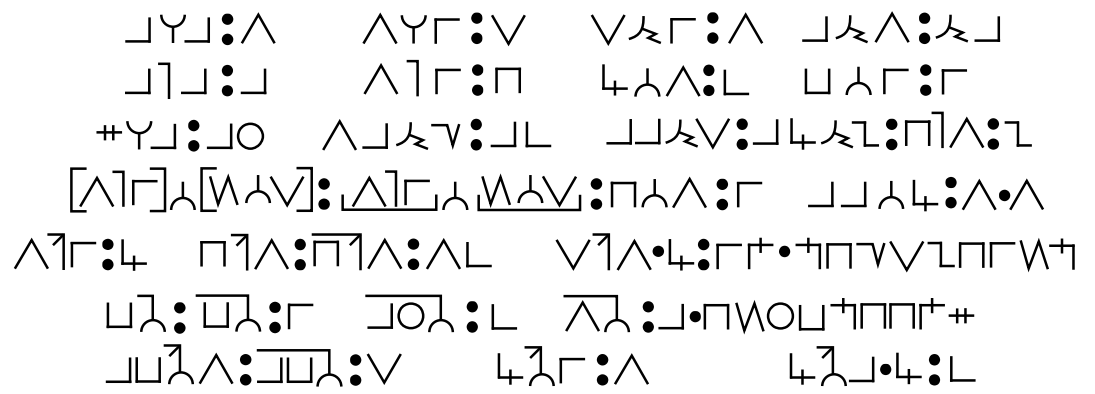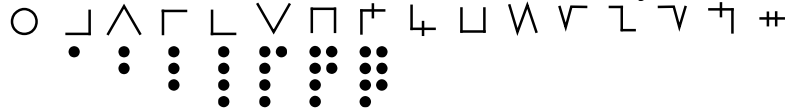
Quick preview for the curious

Here are a few example phrases built with the vocab defined in this document
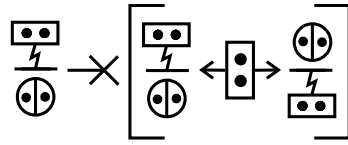
We want to speak with you(plural)
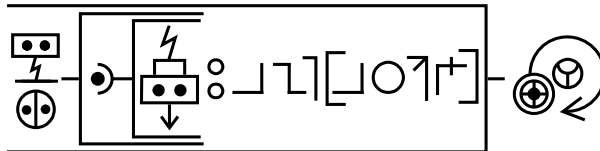
There are approx 7.5 billion of us on a planet
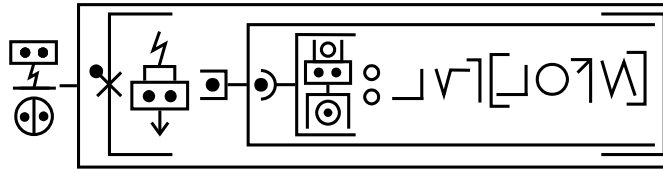
We are each composed of approx 30 trilion unconscious cells
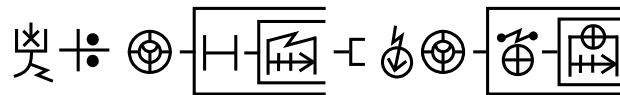
Why is the binding force of glalaxies greater than the gravitational force of the observed mass?

In other words : "what is dark matter?"
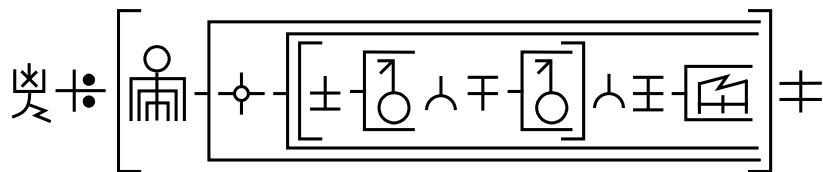
Why is the universe expansion ratio of space divided by elapsed time unchanging?

In other words : "what is dark energy?"

Uscript
A universal logographic physics based language

Uscript (Universal Script) is a language and writing system designed to be as universal as possible. The composition of characters is done similar to Chinese characters:

1. **Logographic**
   Characters represent words / ideas, not sounds / phonetics
2. **Ideogram / Pictogram**
   Attempt to be visual representations of abstract ideas and concrete things
3. **Radical Based**
   There are small fundamental characters which can be combined to form more complex characters.

The language, however, is designed from the ground up, with the following rules:

RULE #1 : Must be universal

"If an alien were to read the message they should be able to understand it without knowing anything about Humankind or Earth. As few assumptions as possible should be made about the life form reading the document. "

RULE #2 : SEE RULE #1

* There are of course several goals in its design like efficiency, optimization for carving/pen/brush, elegance of form and attempt to be intuitive and symbolic. The "rules" above are a joke meant to emphasize that most important is that it divorce itself as much as possible from all human-centric elements.

There are really only 2 possible fields to base a universal language on : Math & physics

Math and physics are deeply intertwined. Physics can be seen as applying math to the physical world. Physical concepts in general are more suitable as morphemes for language, math will be used mostly to help with definitions, and of course express numbers/equations/etc...

A lot of Chemistry, pretty much all biology, and other higher level fields are "not universal enough" because they depend too much on environmental factors that can vary largely between environment. eg. which chemical structures are common and stable depends heavily on temperature, pressure, environmental composition, ambient radiation, etc..

One of the inspirations for Uscript was the Voyager and Pioneer plaques. On these plaques they attempted to inscribe messages for aliens, but in order to convey measurements they had to find a universal unit. Measurements were expressed using units based on the Hydrogen 1 spectral line, hydrogen being the most common element in the universe. The 21cm (1.42 Ghz) Hydrogen spectral line is detectable everywhere so it is rather safe to assume aliens would be familiar with it and it can be visually described via the atomic process that generates it.
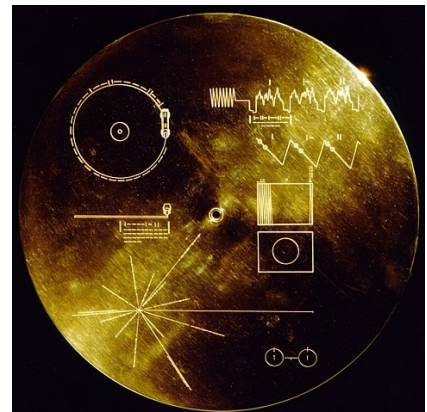
Uscript Numbers

Numbers are of course necessary, they are fundamental to virtually any communication and needed to communicate detailed and well defined descriptions of any kind. Whether you want to communicate scientific concepts or have a basic conversation, numbers are inevitable.
Firstly, Which base should be used?

Base 10   NO!

We use base 10 for an obvious reason, we have 10 biological digits. Other than the fact that we have 10 fingers and 10 toes, there is no good reason to use base 10. I often gripe "Stupid 10 fingers! I wish we had 8, they could be be just as useful and then we would use base 8, making my job as a programmer so much easier"(I bit bang and design custom data/comm formats a lot).

Base 2   Not bad

Base 2 is the most fundamental counting system, so it's not a bad choice. Numbers end up being rather long, but its fair to assume any technologically advanced alien civilization would be well versed in binary and maybe even use it as their default base.

Base 3   Cool, but no.

Base 3 has some good arguments. Ternary (base 3) computers have been built in the past, and some argue that we ended up using binary because binary technology developed first and it was too burdensome to convert manufacturing methods and basic components. Ternary can also represent both sides of AC waves (positive, neutral, negative).Furthermore ternary can be used in a type of number system that can express positive and negative numbers without adding a sign (see Balanced Ternary https://en.wikipedia.org/wiki/Balanced_ternary).

Another argument for ternary is that it can be used in optics by using the polarity of light, but polarity has a full 360 degrees, so with accurate enough sensors this argument could be used for any base system.

Balanced ternary is hard to mentally process for minds not well trained in balanced number systems like us, whereas even a civilization that did use balanced ternary as their basic number system would still be very likely to develop binary and unbalanced number systems.

The fundamental components that make up computers are gates, ternary computers are actually created by using multiple binary gates. A simple way to imagine this idea is a 1-way valve, ternary computers just use 2 separate 1-way valves to divide the water current into 2 separate tracks. So Ternary logic is still based on binary components.

Some analog systems use ternary in a more "pure" way, but they are usually based on inductor windings/transformers and vacuum tubes, which are bulky and not suitable for complex computing compared to microscopic transistor logic (not bad for analog stuff like audio though).

Quantum computing could bring a renaissance of ternary, but it is also likely to bring about many other bases, and still employ base 2, so it is hardly an argument for the universality of base 3.

In short, base 3 is cool, could arguably be a universal concept, but not the best choice here.

Base 4   Getting better

Base 4 is a pretty good choice. It is a power of the most fundamental number system, base 2. The only downside is that it is still a bit small. Large numbers become large in digits quickly and would require a scientific notation system rather early.

Base 5   NO!

The only argument for base 5 that I can think of, besides our 5 fingers which is human centric, is that it can make a balanced base 5 number system. It could represent +2 +1 0 -1 -2, so similar arguments used for base 3 can apply. A neat idea, but definitely not the best choice

Base 8   Good

Base 8 is a good choice. It's a power of 2, values are close the same length (number digits required) as our base 10, It would require scientific notation a bit earlier than base 10. The only reason I lean towards something else is that it is 2^3 (2 to the power of 3), I don't like that 3. Its a bit arbitrary but since 2 to the power of 4 is just as reasonable an option I would prefer that.

Base 12   Interesting.. but no.

Base 12 has some interesting advantages. Base 12 numbers can be divided by 2,3,4,6,etc.. without remainders. Being able to divide by the 3 smallest integer divisors (2,3,and 4) is handy, but it suffers the same problem as base 10, its messy to convert back and forth between binary(and also ternary), the system that technology will rely on.

Base 16   Yes! Yes! Yes!

Base 16 (hexadecimal) is a great choice. It beats base 10 in value to digits ratio (I only compare to base ten as it is our reference point), so it would allow larger numbers before resorting to scientific notation. It is 4 digits of base 2, in other words its is 2^{2^2} (2 to the power of 2 to the power of 2). Base 16 has all the advantage of base 2, its is composed of 4 bits, base 16 logic and technology can use some cool digital hardware optimizations.

Base 256   Too far! Go back!

Base 256 is $2^{2^{2^2}}$, or $(2^2)^{(2^2)}$, it is slightly more "symmetrical" than base 16. We need to be able to write these numbers as symbols, however, and 256 is excessive I think.
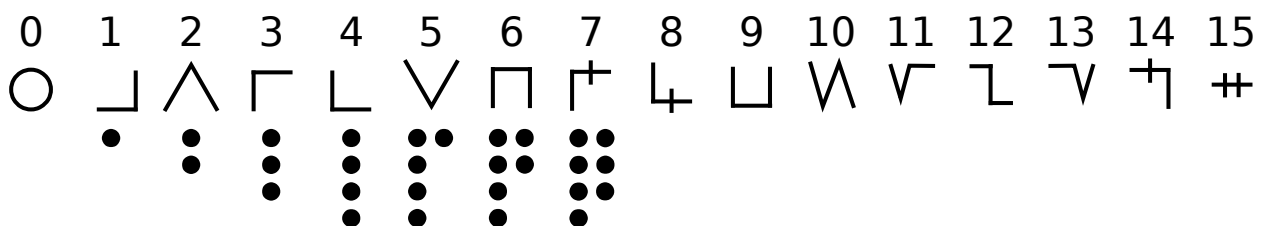
Numbers are perhaps the easiest to define set of symbols. By listing number symbols and associating collections of dots any number can be made universally defined. I don't thinks it is necessary to draw dots for all the symbols, just enough for the pattern of +1 to be plainly obvious.

Reading right → left we place larger powers on the left, so that they are read first, in other words the symbols are little-endian (smallest powers on the right). These number symbols are pictographs of a binary signal (with some leading 0 values removed)

1 = low low low HIGH	represented as	low low HIGH
2 = low low HIGH low	represented as	low HIGH low
3 = low low HIGH HIGH	represented as	low HIGH HIGH
4 = low HIGH low low	represented as	HIGH low low
5 = low HIGH low HIGH	represented as	HIGH low HIGH

etc...

Uscript radicals

        The Uscript radicals were designed to be either relatively obvious visual representation of the concepts they represent and/or be visually defined using more obvious radicals in the same way numbers can be defined using dots to indicated we are defining symbols as numeric values.

        These symbols do not" self-define" alone, they are defined in graphics on following pages and and in further sections. They do no need to be obvious but I try as hard as possible to make them visually representative.

Particle	●
Space	○ ▢
Wave	∿ ⚡
Time	＋
Fork	人 �псой
Merge	Y Y
1D	│
2D	⌐
3D	⏶
Subset	⊓⊢
Superset	⊔⊣
Bond	⊤⊥
Hadron	△
Movement	↓
Container	⎣⎦

There is of course a fundamental assumptions that must be made to read the script. Reading direction in Uscript is top → bottom , left → right, both of these are not universal. We must assume that this will be discovered through analysis, the math and physics examples can help establish this.

It is not impossible to read it backwards or bottom to top and understand it, who knows, perhaps there are other life forms that prefer to read information in a way that we would consider reverse order.

Complex characters are combinations of radicals, these radicals are also read top → bottom and left → right. But the whole symbol is what matters so this is not important.

To clarify this a "movement arrow" can be used to define these reading rules. (The description of the arrow radical explains how and why this can be used in a universal way)

There are reasons for these radical choices, and they are arguably intuitive or able to be determined through context or by analyzing larger quantities of Uscript text.

Some radicals have multiple forms, to make it perfectly clear without needing any guesswork or analysis, simply including the chart on the left (without English) will make clear which radicals have multiple forms and what they are.

The individual radical definitions explain why these radicals can be seen as good visual representation, what they can represent, and how their meaning can defined visually without use of any external language.

Radical introductions

⬤ **Particle, Matter, Point, Radix**

Meaning	Reasoning
Particle	A dot is an obvious symbol to represent a particle.
Matter	Matter is observable as particles, while it can be argued that everything including matter is composed of energy and waves, matter tends to be observable and detectable as a "solid particle" as opposed to non-mattter energy which is observed more as waves. Matter also has gravity, which causes it to accumulate on large scales, making it tend towards more point like collections, and even up to black holes singularities which are believed to be a perfect single point.
Point	A dot can be used to indicate a specific point in a field or region.
Radix Point (aka. Decimal point)	While reading a little-endian number (larger powers to smaller powers), reaching a "particle" can indicate that you have hit the level of smallest whole unit, so everything afterwards is smaller than a whole unit.

◯ ▢ **Space, Region, Zero**

Meaning	Reasoning
Space	An empty circle or square are the most obvious visual representation of "empty space"
Region	Because these symbols encapsulate the empty space they can also represent the concept a region. In many languages the word space is often used as a synonym for region.
Zero	Empty space is probably the best metaphor for the number and concept of zero

〰⚡ **Wave, Energy, Information**

Meaning	Reasoning
Wave	A sine wave is the most ideal wave form, the zig-zag is just easier to draw, but this ease is based on the mechanics of the human hand. The sine wave form can be quickly defined to be equivalent to the zig-zag to prevent confusion that may arise from erroneously assuming they are different symbols.
Energy	Energy exists as waves in fields, it can manifest as particles and mass, but it is more broadly associated as more wave-like in form. Although energy comprises matter, energy in matter form has properties more "point-like" whereas non-matter energy tends towards wave-like properties.
Information	Energy and waves are good metaphors for information, they allow particles to interact across space. A wave has many possible forms, so any one form provides information regarding its origin, whereas a point particle (used to represent matter) has 2 states state (existence/non-existence).

Time / Entropy

The "time / entropy" symbols is the least intuitive of all the base radicals, as such will absolutely require a visual definition. Many composites will be built upon this symbol. It is not impossible to to derive its meaning from context given enough text, but this symbol in particular should defined.
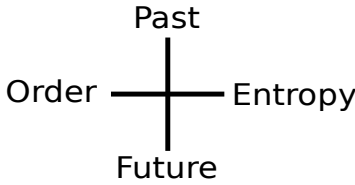
On the left you see the English definition of the time / entropy symbol. Obviously this is just for this document, defining it in a Uscript document must be done visually without reference to other languages. There are many ways this can be done clearly, here are just a few.
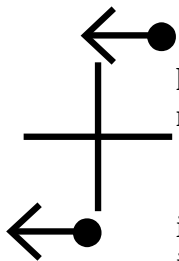
Here on the left you can see using simple particles with movement arrows.We have defined the particle to be moving to the left, and its position at the bottom has moved to the left from the position at the top.

This however is still open to confusion. It is not impossible to see this picture and come up with incorrect but logically sound conclusions. One could logically conclude that the "top of the intersection indicates attraction to center and bottom indicates repulsion from center". This is not what we want so I prefer other visual definitions.
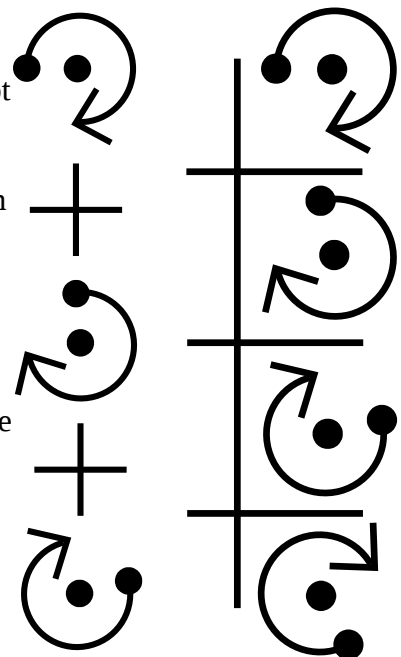
On the right is a better visual definition done in 2 ways. The orbiting particles and direction of of orbit makes it pretty clear that the system is progressing through time. I have included 2 forms of the orbiting time definition to also demonstrate that the time/entropy symbol can be expanded to display multiple "frames". This not only helps the visual definition be clearer, it is also a useful way to apply the time/entropy symbol to express multi-stage process and stages of time-limes.

Entropy is trickier. A popular understanding is that entropy is "chaos/disorder" or "diffusion", but the scientific definition is more nuanced than this. Entropy is only required to always increase in closed systems, and systems can be naively thought to decrease in entropy simply due to a misunderstanding of the nature of entropy and randomness.
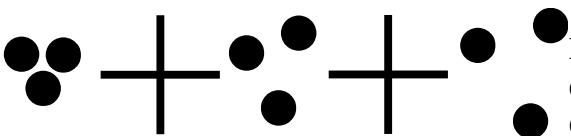
Here on the left you see one simple way to visually describe entropy, To better clarify that we are referring to entropy, and not just just particle diffusion, we can include a couple more visual definitions. Below you see one where a particle is decaying into waves, and another where a wave is red-shifted. These 3 definitions together should be enough to hint toward entropy, or at the very least imply heat-death, but it could still be confused with both axis being time. Entropy will be better defined and distinguished from time later in this document.
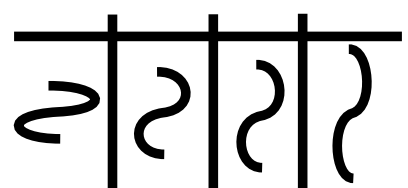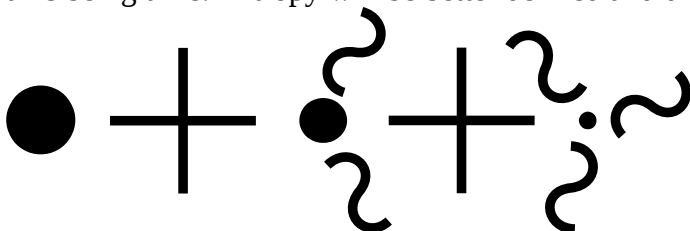
人人 **Fork / Divide**

Meaning	Reasoning
Fork	The fork symbol shows one line splitting into 2, an obvious fork. To clarify it any visual definition showing an object or system being divided can be used.
Divide	A fork is a good visual for division by 2. In order to use it as a general mathematical symbol for the division operator it must be defined as such. Defining it as division will also clarify the fork meaning. This will be shown in the next section on math definitions in this document.

ΥΥ **Merge / Combine / Fuse**

Meaning	Reasoning
Merge	1D lines merging into a single line is a simple obvious visual for merging.
Combine	The concept of merge is easily extended to the concept of combine.
Fuse	Fuse is a rather obvious extension of the concepts merge and combine. A visual definition of two particles combing into a single particle can be used to define it visually.

| **1D / Line**

Meaning	Reasoning
1D	A line is a perfect visualization of 1D.
Line	Nothing could be better to define a line than a line.

⌐ **2D / Plane / Multiply**

Meaning	Reasoning
2D	2 lines is a decent visual metaphor for 2D. It is not clear though, it could be interpreted as other things like "corner". It is defined in the math section.
Plane	The concept of 2D is easily extended to mean plan.
Multiply	Multiply is a good extension of the concept of Plane and 2D. This meaning will definitely require further definition. Proper definition will be done in the next section of this document for math concepts.

⌐ **3D / Exponent**

Meaning	Reasoning
3D	3 Lines in the shape of a cube wireframe corner, good but still requires further definition.
Exponent	3D can be a rather stretched metaphor for exponent, Further definition in the next section of this document on Mathematics.

* All mathematical and dimensional meanings will be elaborated upon in the math section

⊐ **Superset / Greater than**
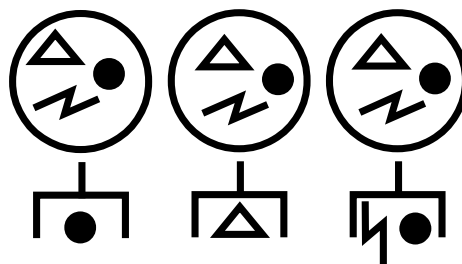
Meaning	Reasoning
Superset	This symbol is a decent representation of superset. Subset and superset only differ in how they are read. Seeing as Uscript is read top → bottom the symbols above would be read X(several symbols in a group) is a superset of Y(one of those symbols).
Greater than	Being a superset of of something implies that it is greater than. Detailed definitions of this will also be included the math section of this document.

⊏ **Subset / Less than**
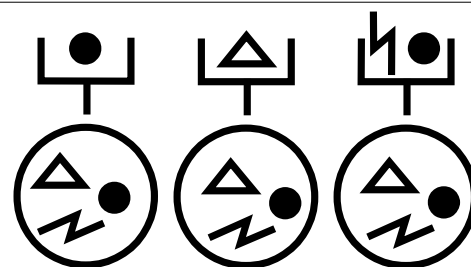
Meaning	Reasoning
Subet	The subset symbol is a vertically inverted superset symbol. The subset symbol is read top → bottom, the symbols above would be read X(a single symbol) is a subset of Y(a group of symbols including that symbol).
Less than	Being a subset of of something implies that it is less than the whole. Detailed definitions of this will also be included the math section of this document.

⊥ **Bond**
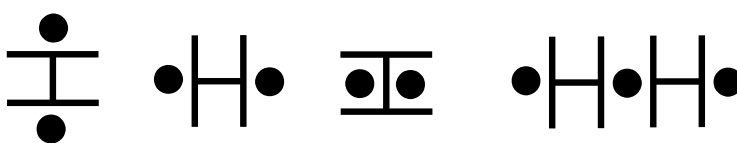
Meaning	Reasoning
Bond	The bond symbol is not very intuitive alone. It can be used in any rotation, with bonded units inside or outside the bond symbol, and used to represent multiple bonds in larger bonded structures. Defining the meaning of this radical will be done in detail later this document.

△ **Hadron / Quark**

Meaning	Reasoning
Hadron / Quark	Hadrons are quarks held together by the strong force which is has three types of charge hence a triangle. This symbol and the concept will be further explained and defined later in this document.

→ **Movement / Momentum / Direction**

Meaning	Reasoning
Movement / Momentum / Direction	An arrow is a great representation of movement and direction. The arrow symbol we use is not arbitrary. Aerodynamics and fluid dynamics in any environment make the arrow the most ideal design for projectiles. The heavy pointed end will always be the direction the projectile moves in. While it is possible to design projectiles shaped more like a rocket where fins are in the back, the fins will generally be pointed backwards. It is safe to assume this shape will be universally understood to mean something relating to direction and movement.

⌐_⌐ **Container**

Meaning	Reasoning
Container	The container symbol is a visual metaphor of a container, it is akin to brackets. An open top box is perhaps not universal as its usefulness depends on aerodynamics/fluid dynamics and gravity, but it can be rotated.

Uscript Math

　　Once you have numbers mathematical symbols can be anything you want and be defined easily by providing some example formulas. The symbols are meant to be physical processes that describe the operation, that way they are well suited to be extended to describe physical processes.

= 　　2 Dots is used to indicate equals

+ 　　The combine symbols is used for addition

- 　　The radiate/emit symbol is used for subtraction
　　　　*radiating is non-commutative, like subtraction, and the best physical process to describe subtraction

x 　　The 2D symbols is used for multiplication

% 　　The fork symbols is used for division
　　　　*If I use this for subtraction then I don't have anything good for division and radiate/emit is actually a better fit for subtraction

∧ 　　The 3D symbol is used for exponentiation
　　　　*3d is a very stretched metaphor here. not ideal I admit

√ 　　A combination of 2D and Fork symbol are used for square root

$\sqrt[n]{}$ 　　3D-fork means Nth root. The N is put after the symbol

** The symbols try to find a physical process or property to match the math operator. Many may find it odd that +/- are not paired. This is the best fit for the set as whole using our fundamental symbols and processes.

Mathematical meanings can be most clearly defined by using some example equations. The meaning of the 1D, 2D, 3D, 4D symbols can be clearly defined with the visual definition on the right. The concept can extend into higher dimensions as well.
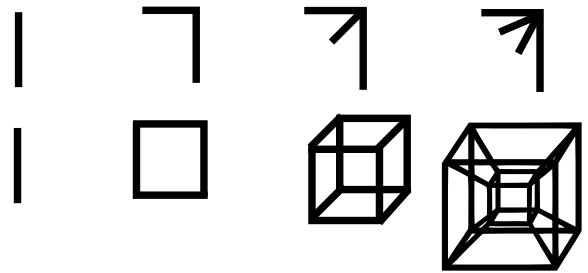
Reasoning for math symbols
- Equals is 2 dots, implying 2 things that are equal
- Merge is an obvious choice for addition
- Radiate for subtraction is like a value radiates away a value
    *radiation process is defined in later section of this document
- 2D is a metaphor for multiplication in that a plane of x length and y width is an area of x*y
- Fork symbol represents divide. Forking is usually more like division than subtraction
- 3D symbol for exponentiation is a bit of a leap. Definition required.
- 2D+fork means square root because a square root can be considered a 2D division in that it finds out what length the side of a square would produce the area equal to the input value.
- 3D+fork means nth root. 2D will be reserved for square root because it is such a common operation and 3D will be used for rooting in general. This is arbitrary but will be easily made clear with a few sample equations

Example equation / Visual definitions

⌐Y⌐:∧   ∧Y⌐:∨   ∨�ㅈ⌐:∧   ⌐ㅈ∧:ㅈ⌐
⌐⌐⌐:⌐   ∧⌐⌐:⊓   ㄴㅅ∧:ㄴ   ⊔ㅅ⌐:⌐
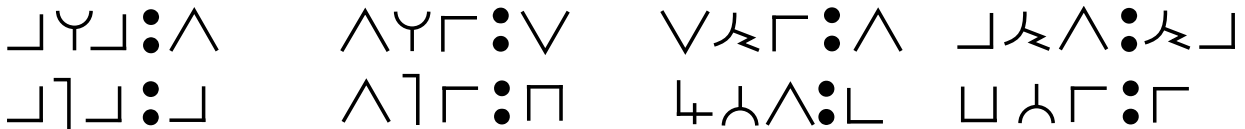
Above are some example equations that clearly define operation using only the numbers already defined

Equations are:
1+1=2	2+3=5	5-3=2	1-2=-1
1x1=1	2x3=6	8/2=4	9/3=3

These basic equations should clearly define +,-,x,% and =, it also gives an example of negative numbers, which just uses the subtract symbol as a prefix like we do normally. Using a negative symbol prefix to indicate negative numbers is a bit arbitrary, there are other possible forms of notation, but other than balanced odd number bases all other choices are just as arbitrary. Being intuitive and clearly definable is all that is needed to be universal, and it meets these criteria.

+Y⌐:⌐O   ∧⌐ㅈ∨:⌐ㄴ   ⌐⌐ㅅ+:∧•∧   ⌐⌐ㅈ∨:⌐+ㅈㄱ:⊓∧:ㄱ
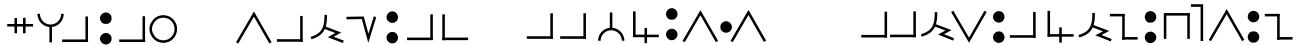
Next we can use some equations to to clarify how the power digits are written. Just like the bits that make up each number symbol, the numbers are little-endian left → right. The above equations are in hexadecimal and decimal(values converted to base 10) :

HEX:	F+1=10	21-D=14	11/8=2.2	11-5=18-C=6x2=12
DEC:	15+1=16	33-13=20	17/8=2.125	17-5=24-12=6x2=12

∧⌐⌐:+   ⊓⌐∧:⊓⌐∧:∧ㄴ   ∨⌐∧•+:⌐⌐•+⌐⊓∨∨ㄱ⌐⌐W+

Next we add exponents, like most math it is easy to define with a few examples.

HEX:	2^3=8	6^2=6^2=24	5^2.8=37.E6D5C63AE
DEC:	2^3=8	6^2=6^2=36	5^2.5=55.9016994375

⊔⅄: ⊔⅄:Γ ⅂○⅄:ㄴ ∧⅄:⅃•Π\\○⊔⁺ㄒΠΠΓ⁺⧻

⅃⊔⅄∧:⅃⊔⅄:∨ ㄴ⅄Γ:∧ ㄴ⅄⅃•ㄴ:ㄴ

Roots are added next, again a few examples will quickly define the symbol and structure of its use.

HEX:	9(sqr root)=9(sqr root)=3	10(sqr root)=4	2(sqr root)=1.6A09E667F
	19(root)2=19(sqr root)=5	8(root)3=2	8(root)1.8=4
DEC:	9(sqr root)=9(sqr root)=3	16(sqr root)=4	2(sqr root)=1.414213562
	25(root)2=25(sqr root)=5	8(root)3=2	8(root)1.5=4

[∧⅂Γ]⅄[\\⅄∨]: ⅃△⅂⅄\\⅄∨:Π⅄∧:Γ

The container radical can be used as brackets, a few examples will easily clarify this usage.

HEX:	(2x3)%(A%5) = (2x3)%(A%5) = 6%2 = 3
DEC:	(2x3)%(10%5) = (2x3)%(10%5) = 6%2 = 3

Now we have =,+,-,*,%,exponents,roots and brackets. We can now express almost any formula.

Next we need variables, this will give us access to algebra. We use alphabetical letters for this normally, but in Uscript we don't have an alphabet.

We could just allow any undefined symbol to mean "variable", but it is better to have a set of symbols predefined for this.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
• Numbers	⅃	∧	Γ	ㄴ	∨	Π	Γ⁺	ㄴ	⊔	\\	∨⁻	⅂	∇	⁺⅂	⧻
⤳ Variables	⤳⅃	\\	⁀Γ	⤴ㄴ	∨⁻	⁀Π	⁀Γ⁺	⤴ㄴ	⤴⊔	\\\\	∨⁻⁻	⁀⅂	⁀∇	⁺⁀⅂	⤳⧻

By adding a "wave" radical to the first line of each number symbol we create a set of 15 symbols. It can be interpreted as "wave number" = "variable". This needs to be defined of course, so we can do that with a few quick examples.

⤳⅃⅄\\:Γ \\:Γ⤸⤳⅃ [∧⅂]⤳⅃⅄\\:Γ⅄⤳⅃

[Γ⅄⤳⅃:ㄴ] : [⤳⅃:ㄴ⤸Γ] : [⤳⅃:∨]

Above we have a few algebra examples. These 3 examples should be enough to get the message across. We will assume we don't need to explain what a variable is and the concept should be universal and recognizable, a safe bet for any technological civilization. But we may need to provide examples of solving algebraic equations, solving the numerical value of a variable will ensure that variables are not just confused for an alternative writing form for numbers.

The equations are : a + b = c   b = c – a   (2a) + b = c + a
(3+a=8) = (a=8-3) = (a=5)

Variables will be used in many universal laws and equations so it will defined even better later.

At this point we get to the real of advanced math(s). Trig...Calculus... and beyond. This I will leave mostly open for now. I do not wish to build too many structures for mathematical expressions before getting to the more linguistic elements, I fear restricting available designs for language too much. So these will require a second pass.

Constants and concepts

There a tons of mathematical concepts and constants that we can define. Here are a few basic ones. Math is an insanely deep field and I'm only going to build a few things we may need right away. I'm just scratching some surface here.. and we expand a bit later... this rabbit hole be deep.

⊕ Diameter   ⟳ Circumference

⊙ Radius   ⬭ PI   ⯑ e   2D Particle i (√-1)

1D Space Distance   Circle Diameter   Plane X   Cuboid X

2D Space Area   Sphere Diameter   Plane Y   Cuboid Y

3D Space Volume   Cuboid Z

Triangle   Right Triangle   Sine

Triangle Side   Trig Side   Cosine

Triangle Angle   Trig Angle   Tangent

Hypotenuse

Untranslated lower portions reads:

pi = 3.243F6(hex)   e = 2.B7E15(hex)   i = sqr root of -1

radius + radius = diameter   circumference = pi * diameter   circle area = pi * (radius pwr 2)

sphere volume = (4/3) * pi * (radius pwr 3)   area of 3d sphere = 4 * pi * (radius pwr 2)

rectangle area = x * y   cubic space = x * y * z   surface of cubic space = 2 * ((x*y) + (y*z) + (x*z))

sum of 3 angles of a triangle = 1/2 circumference   sum 2 non-right angles of a right triangle = 1/4 circumference

sine(angle) = opposite / hypotenuse   cosine (angle) = adjacent / hypotenuse   tan(angle) = opposite / adjacent

circumference is used to denote a full circular revolution, one circumference is 360 degrees

Uscript Physics

Now that we have some basic math we proceed to defining "things" and "processes", in other words "nouns" and "verbs". (technically we already have a few, number are nouns and operations are verbs)

We will start with fundamental universal physics like particles, forces and interactions.

Higher level language will have to be built upon these. For example if you were to jump straight from physics to conversation then "I like you" would have to be expressed as something akin to "particle emitting this wave(I) attracted to(like) particle absorbing this wave(you)".

This may seem strange or cumbersome, but it is necessary to ensure maximum universality. No human and earth centric concepts may be used. In reality there are many levels of definitions between physics and "I like you" so it will quite different form the above example.

Particles

The simplest universal way to define the elementary particles is by mass. For this we first need a mass unit. The best solution for this is to define all the particles in relation to one of the particles. So which particle do we choose?

The most stable and common particles in the universe are photons, electrons and protons. Photons have no mass, protons are quark composites, so that leaves us with electrons as the obvious choice.

So now that we have our unit of mass we can define all the elementary particles… the only exception is that the gluon and photon both have 0 mass, so they will need to be distinguished from each other via other means.

Fermions

Leptons			Quarks		
electron 0.511 MeV/c²	muon 105.6 MeV/c²	tao 1.776 GeV/c²	up 2.2 MeV/c²	charm 1.28 GeV/c²	top 173.1 GeV/c²
electron neutrino 2.2 eV/c²	muon neutrino 0.17 MeV/c²	tao neutrino 18.2 MeV/c²	down 4.7 MeV/c²	strange 96 MeV/c²	bottom 4.18 GeV/c²

Bosons

Z boson	W boson	higgs	photon	gluon
91.19 GeV/c²	80.39 GeV/c²	124.9 GeV/c²	0 GeV/c²	0 GeV/c²

We will need to convert all values into electron mass units (1=the mass of an electron), so we will divide all values by the mass of an electron, and convert into hexadecimal.

The precision of the mass values of particles is not perfect, except for the electron and muon for which we have relatively precise values, most values are not very precise yet, but we will make do

with what we have for now. These values can always be updated as our precision increases. This precision should be plenty to distinguish the particles from each other and the ratios between them should be unambiguously identifiable as the elementary particle mass values.

Even if a civilization has not yet resolved all of these values, with enough of them they should be able to deduce the meaning of unknown values and particles.

	MeV/c^2	Electron Mass Units	Hexadecimal
Electron	0.511	1	1
Muon	105.6583	206.7677	CE.C487FC
Tau	1,776	3475.538	D93.89BA5
Electron neutrino	0.0000022	0.000004305	0.00004839D
Muon neutrino	0.170	0.33268	0.552A8438
Tau neutrino	15.5	30.33268	1E.552A843
Up quark	2.2	4.30528	4.4E26D480
Charm quark	1280	2327.27	917.451EB8
Top quark	173,000	338,551	52A77
Down quark	4.6	9.001	9.00068
Strange quark	96	187.866	BB.DDB22D
Bottom quark	4180	8180.039	1FF4.09FB
Z boson	91,180	178454	2B916
W boson	80,380	157299	26673
Higgs boson	125,000	244618	3BB8A

And next we use these values to associate symbols with each particle. First I would like to create a character for mass, a simple chart with symbols and numeric mass values would suffice, but a character for mass is rather vital so I will define it here.
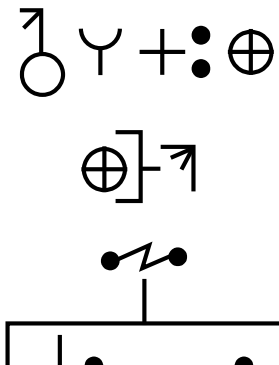
On the right first is the definition for space time, and below it a definition for interactions.

Our character for mass will read as "interact space-time"

The definition for space-time is

3D space + Time = (new symbol) space-time

The next line simply states that space-time is 4D

Space-time is a subset of 4D

Finally there is a visual definition of interactions. It shows that the symbol presented contains a subset that involves 2 particles, over time one particle radiates a wave, and the other absorbs it.
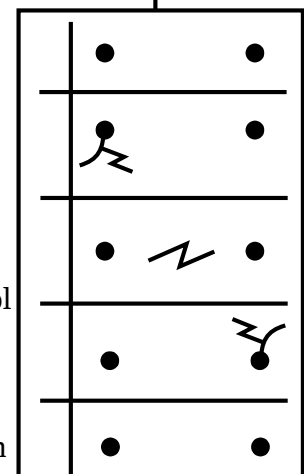
It only defines this radiate-absorb interaction as a subset of the interaction symbol, this way "interaction" can still have a broader meaning.
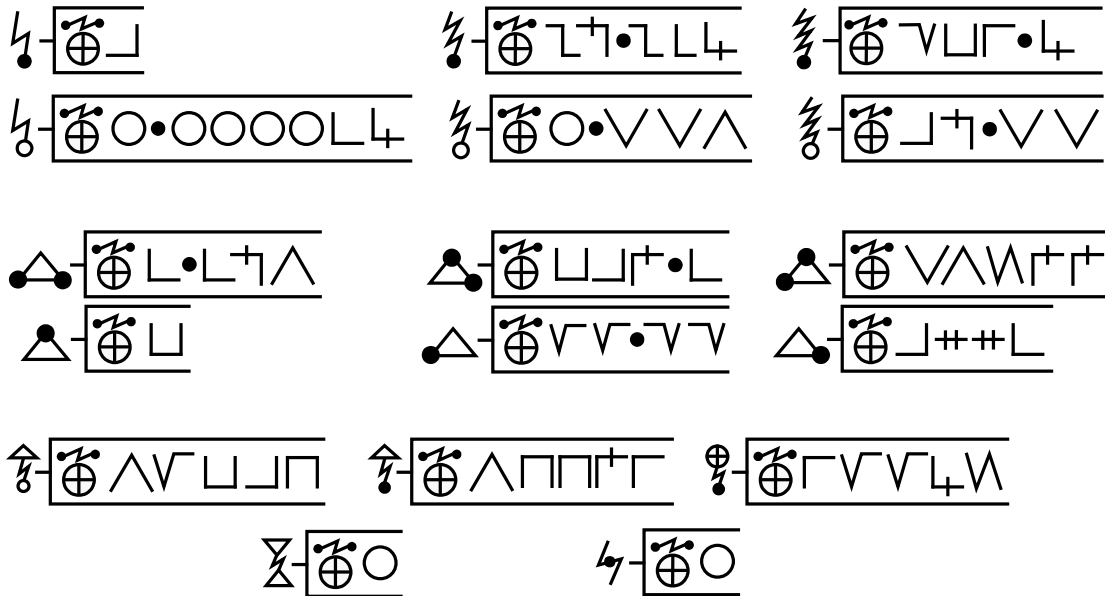
*some of the resolution in my numbers not real and simply a product of unit and base conversion

Above we have defined all the elementary particles by mass. Now we can use the particles symbols to quickly define other properties like charge and spin, as well as differentiate gluons and photons by describing their interactions between particles.

It's about time for words like "or" "and" etc..

We will build binary logical gates and use them as their linguistic equivalents, as opposed to naming the gates after their linguistic equivalent as we normally do.

The left of the image uses a truth table to define a simple system for logic gates. We use an X shape and mark or the TRUE quadrants by placing dots on the respective "X terminals"
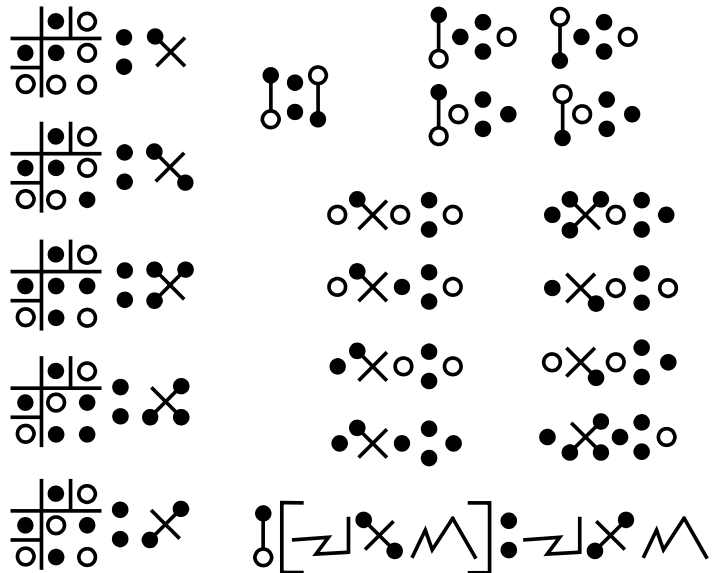
The image on the left shows the following

Far left (top to bottom) AND, Inverted XOR, OR, NAND, XOR.

Next near top middle we define the inversion symbol as having 2 forms, then to the left of that we have 2 examples of each inversion symbol. They both operate the same of course, just 2 different drawing methods.

Below that there are 8 logic examples

Column 1:	0 AND 0 = 0	0 AND 1 = 0	1 AND 0 =0	1 AND 1 = 1
Column 2:	1 OR 0 =1	1 NOR 0 = 0	0 NOR 0 = 1	1 NAND 1 = 0

Finally at the bottom right is the statement invert(a (inverted XOR) b) = A XOR B
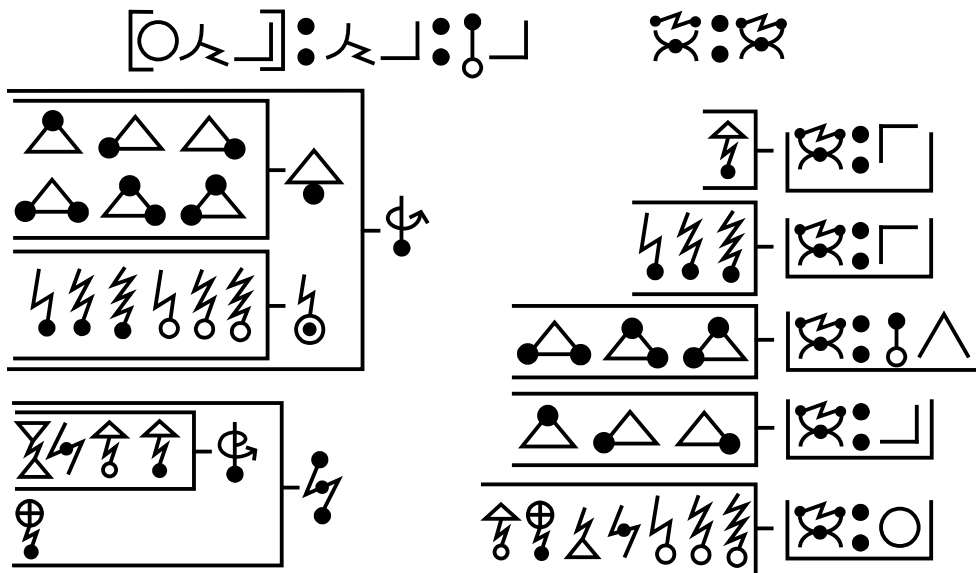
In the above graphic we first create a new symbol for negative numbers. Using the same symbol for subtract and negative makes sense in math perhaps, but not in physics. We really need to get away from expressing negative values as "subtract x", this is NOT good idea in my opinion, it can confuse the process of "radiate/emit/subtract" with "negative/opposite" but we will allow both.

Top left we define that the invert symbol can represent negative values
         0-1 = -1 = invert 1
On the middle left we create symbols for Quark, Lepton, and Fermion.
On the bottom left we create the symbols for Gauge Bosons, and Bosons

In the top right we define a new symbols for "electric charge". We reuse the "interaction symbol" and add a new symbol (the bottom half of the symbol) for "EM field". The "EM field" portion has not yet been defined, however simply using it here will make it clear enough what it means and the context of the mass symbol already defined will allow us to consider the "charge" symbol clearly defined from here on (The full concept of "EM field" will require further definition). A second version of the symbol is defined as well, one that simply merges the 2 components into a single symbol.

On the right we assign charge values to the particles. All charge values are triple what we are used to. Multiplying all the charge values by 3 was done because I wanted to make all particles charges integers. I realize this is going to irk some physicists, I am not married to the idea, natural units could be used it would just make the chart a bit messier. I thought it might be fun to try this charge unit and see how it works. In later documents when I get to describing more complex physics equations I am curious to see what effect it has as I have never seen this charge unit system used.

I do feel a bit of remorse having based mass on electrons and charge on quarks, and not using a natural unit charge. I suppose I could base mass on the up quark but that would only help 1/6 quarks become integer mass values, no real gain. I prefer a smaller value because the electron neutrino has such a small value and I can just barely avoid using scientific notation in these charts.

The symbol we define here for W boson is specifically the W- boson. The W+ boson will be represented as an anti-particle.

*Of course a unit system akin to Kilo, Mega, Giga, etc.. could be devised, and scientific notation can obviously be used since we have all the math required, but I managed to avoid them so far without resorting to long numbers and shall continue hold out as long as reasonably possible.

Some particle symbols reasonings (I tried as hard as possible to not just assign arbitrary symbols)
In almost all cases DOTS represent charge and CIRCLES represent lack of charge.

Symbols for spin particle categories combine (1D + particle + rotation movement) thus creating symbols we can use for angular momentum as well (just remove the particle DOT).

Gluons are strong force carriers for Quarks, and W/Z bosons are responsible for weak decay which allows quarks to change into other types of quarks, so these bosons include the hadron/quark triangle.

Leptons increase their waves with increasing mass.

The Higgs Bosons starts with "Space-time" because in gives mass which bends space-time.

The photon is symmetrical because it is its own anti-particle.

The symbol for "Boson" is based on the symbol for interaction because they carry the forces.

Anti-Particles
Particles with anti-particles have been designed so they can be vertically inverted to represent their anti-particle compliments.

For bosons with no meaningful concept of anti-particle I tried to make them vertically symmetrical if it could be done without much sacrifice to what I consider "ideal representative symbol design".
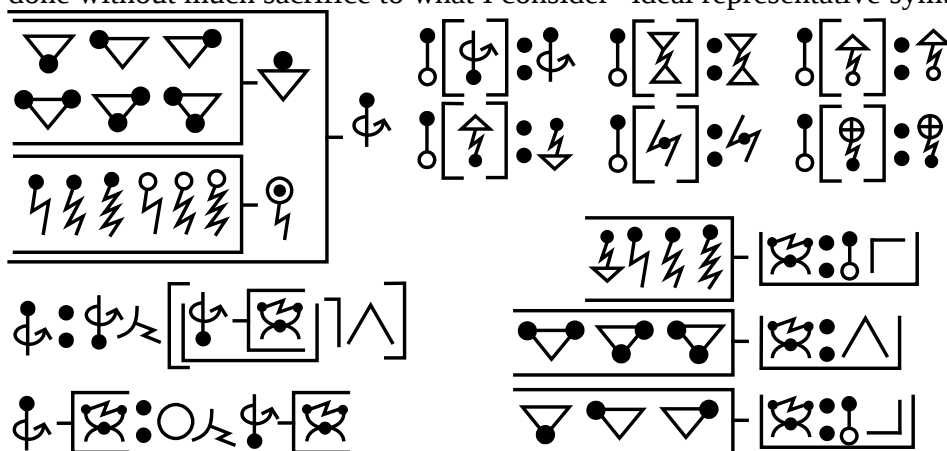
The image above defines and explains symbols for anti-particles.

Top Left : Define symbols anti-fermion, anti-quark (and each individual anti-quark) as well as anti-lepton(and each individual anti-lepton).

Bottom Left : 2 statements to define anti-fermions and clarify some methods of applying expressions regarding particle properties.
      anti-fermion = fermion - ([fermion charge] * 2)
      anti-fermion charge = 0 – fermion charge

Top Right : invert(fermion) = anti-fermion   invert(gluon) = gluon   invert(Z boson) = Z boson
      invert(W- boson) = W+ boson   invert(photon) = photon   invert(higgs) = higgs

Bottom Right : We define the charge of each anti-particle with charge. This is redundant as we have already defined it with the expressions on the Bottom Left, but this can help ensure clarity.

Hadrons and Atoms

Now that we have elementary particles we can define protons, neutrons and atoms. We already have universal units for mass and electric charge, using the same method as the voyager and pioneer plaques we can use the hydrogen line to give universal units of time and distance. Once we add time and distance we can build composite units like speed, area, volume, density, force, power, etc...

On the right are some very quick visual definitions to give us symbols for proton, neutron and atom. We define the atom symbol with examples as subsets because there are many types of atoms.

Technically we should include strong force bonds and EM bonds in these definitions, but these are still pretty clear without them and I am going for a minimalistic approach for now, defining only as many symbols as needed to reach each next level. Using brackets, even with the subset container should help clarify we mean that these particles are are combined into a single unit. Later the bonding forces can all be added but it is important to leave as much flexibility as possible for higher levels. (I am writing this document as I design the script. This pdf not only documents the script but also the creative design process.)
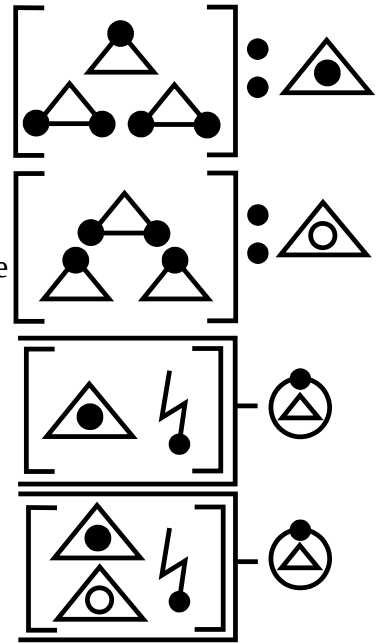
Isotopes
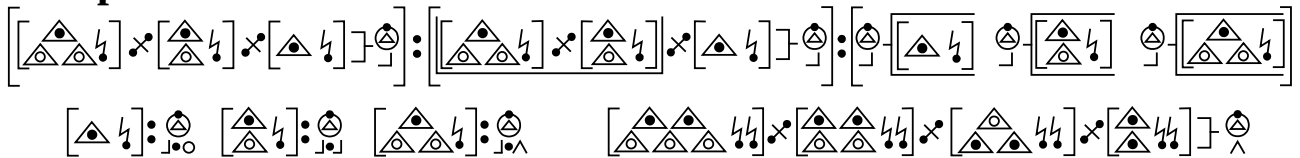
Above we define a notation for atom types and isotopes
The top line in the graphic above reads:

   { (2N 1P 1E) xor (1N 1P 1E) xor (1P 1E) Subset-Symbol Atom-1 }
  ={ [(2N 1P 1E) xor (1N 1P 1E)] xor (1P 1E) =Subset-Symbol Atom-1 }
  =[Atom-1 has subset (1P 1E), Atom-1 has subset (1N 1P 1E), Atom-1 has subset (2N 1P 1E)]

This defines that Atom-1 is Hydrogen, including all its isotopes.
It also clarifies a usage of XOR, and that successive XORs means only one of the many, and that the subset symbol can be used without encasing the subset expression.
        this usage structure of subset may get deprecated

This new way of using subset takes on a more natural linguist role as a standalone "is/are" symbol.
        A is B (A subset B), does not necessarily also imply B is A
        whereas
        A = B implies B = A

The next line defines notation for isotopes and defines the Helium(Atom-2) atom/isotopes to clarify how the notation system scales to all atoms/isotopes. It reads:

        (1P 1E) = Atom-1.0   (1N 1P 1E)=Atom-1.1   (2N 1P 1E)=Atom 1.2
        (3N 2P 2E) xor (2N 2P 2E) xor (1N 2P 2E) xor (2P 2E) = Atom-2

Ions

Next we should define Ions.
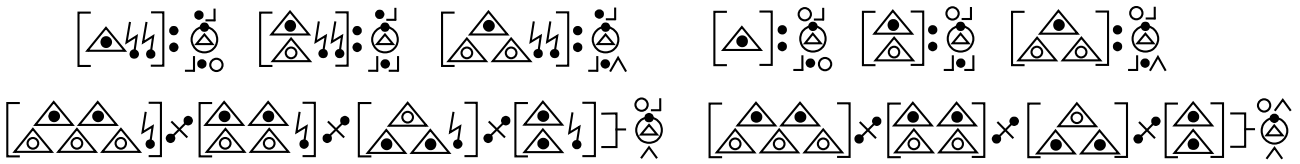
Above we have a simple notation for anion and cations. Ionic charge goes above the atom symbol, whereas information about the nucleus is below the atom symbol. This seems appropriate as the electron shell is much more important in determining interactions and properties.

The top line reads:

(1P 2E)	=	+1-Atom-1.0
(1P 1N 2E)	=	+1-Atom-1.1
(1P 2N 2E)	=	+1-Atom-1.2
(1P)	=	-1-Atom-1.0
(1P 1N)	=	-1-Atom-1.1
(1P 2N)	=	-1-Atom-1.2

The bottom line contains 2 larger statements. This help clarify that it extends to stronger charges and other atoms. The statements are:

(2P 3N 1E) xor (2P 2N 1E) xor (2P 1N 1E) xor (2P 1E) subset -1-Atom-2

(2P 3N) xor (2P 2N) xor (2P 1N) xor (2P) subset -2-Atom-2

*The dot/circle symbols are used to represent charge. Note that dot does not represent negative(positive in Uscript) charge, it just means "charge". The circle (space/zero) symbol is not meant to mean positive(negative charge in Uscript) charge, rather it represents a "lack of/missing charge"

Distance & Time

We will use the Hydrogen 1 line to derive universal units of distance and time.
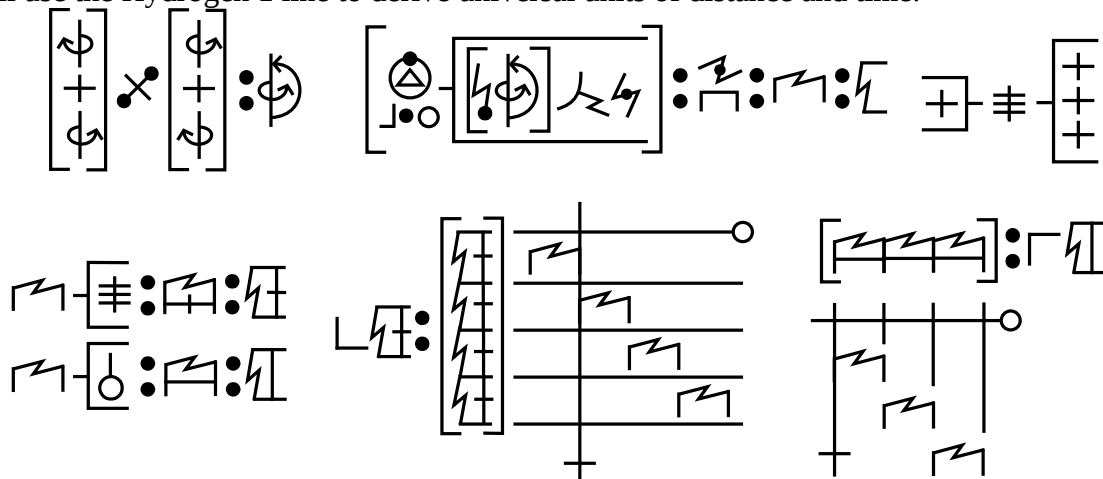
Top Left we define a symbol for "spin flip"

Top Middle :Neutral Hydrogen sub(electron spin flip radiate photon) = photon container = photon unit = photon unit

Top Right : we define a general symbol for "time" to help separate time from entropy and clarify references to time

Bottom Left we define symbols for our units

  photon container sub(time) = photon time unit = photon time unit

  photon container sub (1d space/distance) = photon distance unit = photon distance unit

Bottom right : we use a couple space/time graphs to ensure the symbols are well defined.

Base Units

Unit inventory time. SI base units:

Length	based on hydrogen line
Time	based on hydrogen line
Mass	based on electron mass
Current	can derive, we already have electric charge unit (equiv coulombs)
Mole	arbitrary number for "particle count"
	*Mole not even necessary, can just say "x particles of y"
Candela	Can define arbitrarily based on photon frequency and watts
Temperature	No unit yet, will use triple point of water

So far we have everything we need for all the SI base units except temperature which will be defined using the triple point of water, just like the Kelvin scale, except it wont set the triple point to the human centric 273.16. I will add temperature later, but we will first need molecules and states of matter for that (solid, liquid, gas, plasma)

Since we are comfortable in knowing we can describe the entirety of SI units in Uscript, let's define a few that we will need soon and move on for now.

$$d / t = v \qquad d / (t\verb|^|2) = a \qquad m * v = p \qquad m * a = F$$

Above we define symbols and unit values for velocity, acceleration, momentum, and force.

Gravity
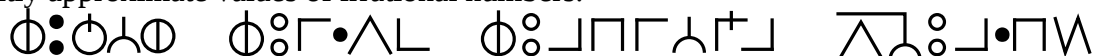
Gravity is a relatively easy concept to define now that we have mass, distance, and force. Well easy if we use Newton's equation, the General Relativity equations are of course much more complex. Newton's equation is usually considered accurate enough for many applications.

General Relativity brings with it the need for a large amount of definitions. While it would of course be possible to define it all right now, I don't want to overload on math yet. "programing / algorithms" is covered later, they will be needed to express GR, as well as be key elements for much linguistic expression. I am aiming to establish a small set of definitions that can act as "minimal key", so GR will be considered part of a "extended information database" as opposed to a fundamental element of the linguistic key.

* "key" refers to the purely Uscript based diagrams that "self define" and establish a language. eg. The math section is a "stand alone key", whereas the physics section requires many elements of math, "math section + physics section" is a valid key but the physics section without any math/numbers could never define itself properly. Oh, you could try, but you would effectively/necessarily create a numeric system and math functions in the process)

Seeing as we will define gravity with an equation that could be considered an approximation, we should add a symbol for approximation.

The concept of approximation should be easy, irrational numbers like pi or square root of 2 should be universal enough, any civilization with math will have come to terms with the fact that they must frequently approximate values of irrational numbers.

Above we have the following statements which will clearly define our symbols for approximation

　　　pi = circumference / diameter　　　pi≈3.24(hex)　　pi≈163/71(hex)　　√2≈1.6A(hex)

*this will be added early in keys to the math section, and the ≈ will be used were appropriate, eg. particle mass values. It is also very useful linguistically.

Now we can define a symbol for gravity without the physics equivalent of grammar police screaming at me or using the "heavy" Einstein field equations.

Above we have the beginnings of the Newtons equation for gravity using variables to reference 2 "things" and a usage of the distance unit that allows us express distances between 2 variables. The equation reads:

　　((variable a subset mass) * (variable b subset mass)) / (a distance unit b)

This is where I have to put in some tedious maths work.. The gravitational constant needs to be used, but it must be converted for our units system. Our current units are :

　　mass : 9.109×10^{-31} Kg
　　distance : 0.211061140542 m
　　time : 0.7040241837 nanoseconds
　　force : $(9.109 \times 10^{-31}$ Kg) $*(0.211061140542$ m) $* ((0.7040241837$ nanoseconds$)^2)$

What have I gotten myself into here *facepalm*. Units are all arbitrary anyways, so these units are just as "good" as SI units, but converting stuff, like the gravitational constant, is gonna be a bit of a chore, and I will have to be very careful or one mistake could get compounded as we build levels of equations and units upon equations and units. Its just a painful conversion, a small price to pay for universal units.

Nothing left to do but roll up our sleeves and get to the math(s). Using Newton's equation:

　　$1 N = 1 kg * 1m / s^2$
　　$(G (1 Kg * 1Kg)) / 1 Meter^2 = G$ Newtons
Uscript units
　　1 Newton = (1Kg-uscript units * 1 Meter-Uscript units) / (1 second-Uscript units)^2
　　$1 N = (1.0977683828808E+30 * 4.7379635940184) / (1420405751.7667^2)$
　　$1 N = 5201186632753682179840206720000 / (1420405751.7667^2)$
　　1 N = 2577968421466.6089985935332 Uscript Force Units

　　(Uscript-G (1Kg-uscript units * 1Kg-Uscript units)) / 1 Meter-Uscript units ^{2} = G N
　　(Uscript-G (1.097768382880E+30 * 1.097768382880E+30)) / 4.7379635940184^2=G N
　　(Uscript G * 1.205095422450970277094400E+60) / 22.4482990182437538962595 = G N

　　UscriptG * 53683150846823097138415015084817964317654304079542732686661 =G N
　　(UscriptG * 53683150846823097138415015084817964317654304079542732686661
　　　　= 2577968421466.6089985935332 Uscript Force Units * G
　　(UscriptG * 53683150846823097138415015084817964317654304079542732686661
　　　　=171.950493711822820206188866444

　　UscriptG= 171.950493711822820206188866444
　　　　　　　　/
　　　　53683150846823097138415015084817964317654304079542732686661

UscriptG = 0.00000000000000000000000000000000000000000000000000000000032030626183336027160567780
　　UscriptG = 3.20306261833360027160567780E-57
This should gives a Uscript-G value for newtons equation of gravitational force. A quick check to make sure we got this right.

Lets do 1Kg under earths gravity at the surface of the earth

Distance from Earths core at surface	6.38E+6 m
Mass of Earth	5.98E+24 kg
Newtons G	6.67E-11

Gravity on 1Kg at surface of Earth
= (1* 6.67E-11 * 5.98E+24)/(6.38E+6 ^ 2)
= 398866000000000 / 40704400000000
= 9.79908805927614705 Newtons

Ok.. we didn't screw that up, now in Uscript units
=(1kg * UscriptG * 5.98E+24 Kg) / (6.38E+6 Meters ^ 2)
=(1.097E+30 * 3.203E-57 * 6.564E+54) / (30228207.729837392 ^ 2)
=2.306E+28 / 9.137E+14

=2.523E+13 Uscript Force Units
= 2.523E+13 / 2.577E+12 Newtons
= 9.79045401629802 Newtons

It works, we now have a functional Uscript gravitational constant.

For notation we can just introduce hex scientific notation, no need for definitions, we have all the math already. I will not establish a a symbol for the constant G, no need for excessive symbols this early, G isn't used much outside the equation for gravitational force anyways, so we just need a symbol for the force itself.

At this point I just realized I didn't define examples of negative powers, but it is not necessary for scientific notation, negative power scientific notation can just be divided instead of multiplied.

3.2030626183336002716056778 0E-57 (DEC)= 1.41B1F88B1D157F61294AE-2F (HEX)
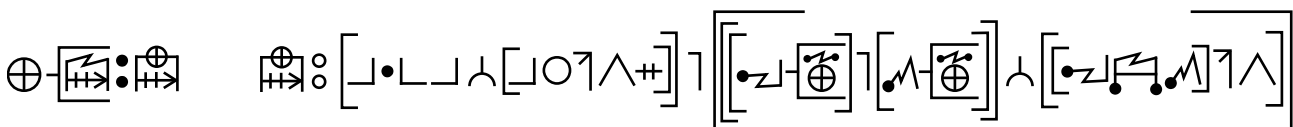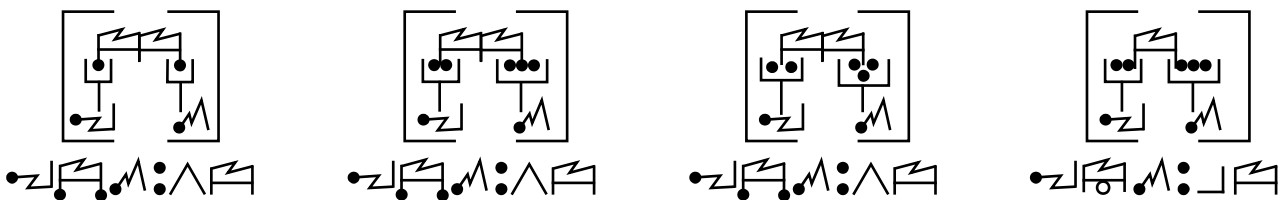
First on top we define symbols for measuring distance. One for distance between "center of mass" and one for "space between". We didn't actually define "center of mass" explicitly, so it could be "spacial center", "center of charge", etc… It is definitely good enough as a general "distance between centers" for now, specific types of "centers" can be further defined as needed. We also start using a new type of variable symbol, the "particle variable", which is just a variable with a dot at its beginning.

Bottom left we define a symbols for gravity as : space-time subset(force) = gravity
Next we define the value of this force using Newtons equation for gravity
gravity ≈ (1.41 / (10^2f))*(((variable A subset mass)*(variable B subset mass)) / (A distance unit B))
*all values in hex. eg.10^2f(hex) = 16^47(dec)

The Weak Force + Feynman Diagrams

I can already hear the physicists thinking "Feynman diagrams!!! why aren't you using Feynman diagrams????" … don't worry, here they come. I have considered what to do with Feynman diagrams and have come to 2 conclusions:

      1. They are PERFECT just they way they are.

      2. The only modifications are purely "cosmetic" for uniformity of the script.
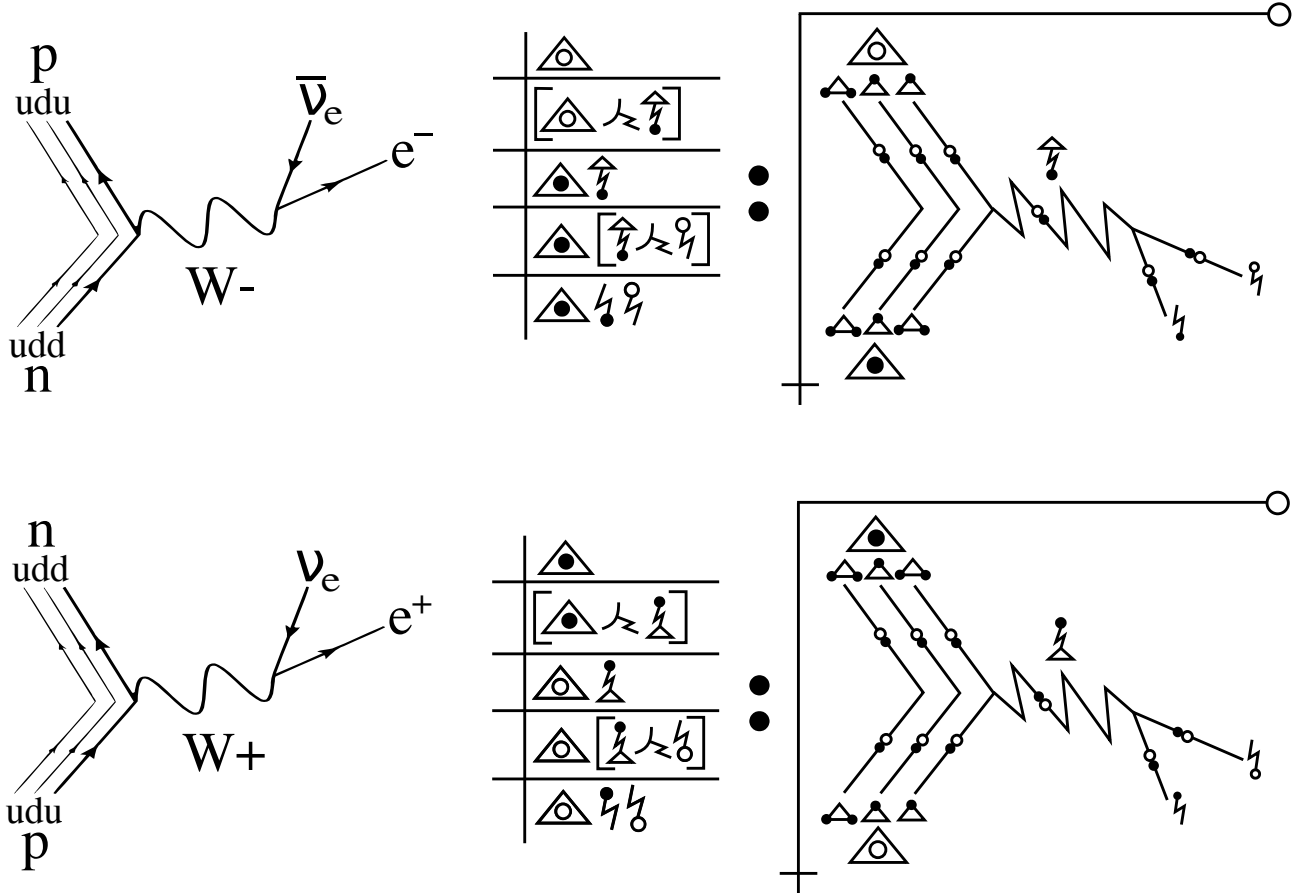
Above we use a couple Feynman diagrams of the weak force in action to accomplish a few things.

      1. Define a Uscript version of Feynman diagrams.

      2. Describe the weak force

      3. Establish that on 2 axis space/time diagrams the intersection in the top left is not needed if you label each axis, one space, one time.

Remember time in Uscript always flows DOWN and RIGHT, not up. We have established that we will be reading from top to bottom and left to right so it only follows that time charts should flow in the same direction.

The 4 Forces

Now that we have gravity defined, we can create symbols to represent the 4 fundamental forces. The equations for the weak, strong, and electromagnetic forces are pretty mathematically intense. They would require a ton of definitions just to be able to describe them. Like Einsteins field equations, we will leave those for an "extended information database" and will not include them in this "core language key".

The 4 forces can be described by the bosons that carry them, and by their relative strength.
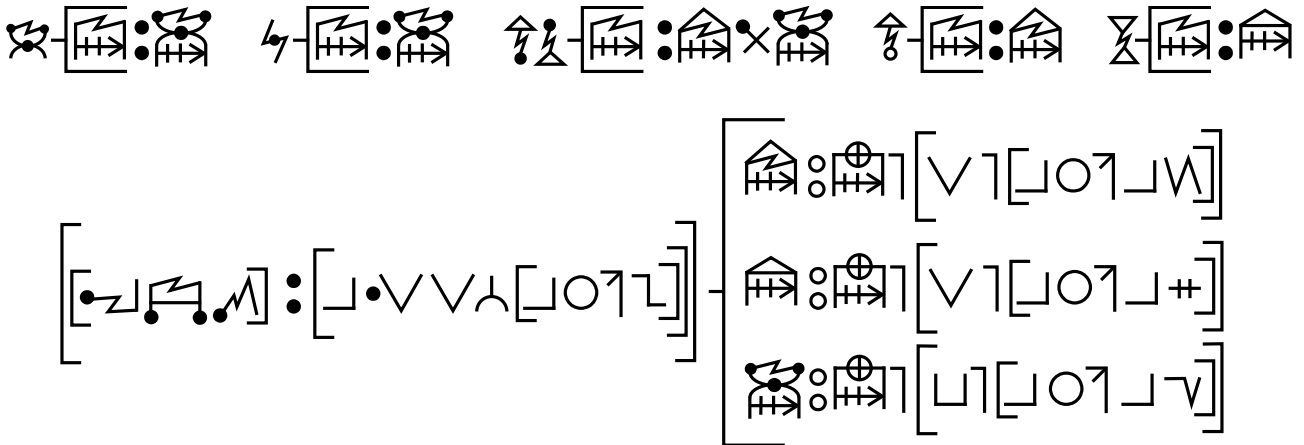
Above top creates symbols for the forces via the force carrying bosons

charge sub (force)	= electromagnetic force
photon sub(force)	= electromagnetic force
W+ / W- sub(force)	= weak force AND electromagnetic force
Z boson sub(force)	= weak force
gluon sub(force)	= strong force

*there is no Higgs boson listed here because the higgs does not "carry gravity"

Below we have a statement that compares approximate relative strength

*from here on in I will use the standard 0x prefix on hex numbers. Easier for everyone, those not familiar with it should just get used to it. 0x10 = 16(decimal). This way you can copy these formulas into calculator/math software and it should recognize it. As a programmer it was hard to resist this long without using it but was trying to not scare some people off, if you made it this far I'm sure you can handle it.

(particle A center-distance particle B) : (0x1.55 / (0x10 ^ 0x12))
            subset - weak force   ≈ gravity * (0x5 * (0x10 ^ 0x1A))
                       - strong force ≈ gravity * (0x5 * (0x10 ^ 0x1F))
                       - EM force   ≈ gravity * (0x9 * (0x10 ^ 0x1D))

Which roughly comes out 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 necesarry

Chromodynamics

For chromodynamics first and foremost.. NO actual color charts, or color metaphors. We have 3 types of retinal color rods, some animals have none, some have less, some have more. Color being "composed of R G and B" is a purely human construct and meaningless to any animals with different sensory configurations. Secondly I want to keep it monochrome, so it can be carved and also read via other senses eg. sonic or tactile.

The easiest way to define some basics of chromodynamics is probably a simple example. We can just use the basic model for Intra-Neutron interactions. This will allow us to define a method for expressing the color charge values of quarks and gluons.

On the right there is a time-line of states which demonstrates the strong force interaction inside a neutron.

I have chosen to use the symbol for "absorb"(inverted "emit") instead of "combine/merge". Seeing as Quarks have mass and gluons don't this is a good way to help clarify a difference between "absorb" and "merge". We will use absorb when there is a merger involving 2 particles of significantly different mass, the more massive particle absorbs the less massive particle, or the particle absorbs the wave.

*here we are describing the effective color of the gluon. Properly discussing gluons is much more complex and delves into group theory. I shall skip this for now. Any civilization that comprehends gluons to a deeper degree will also understand this model.
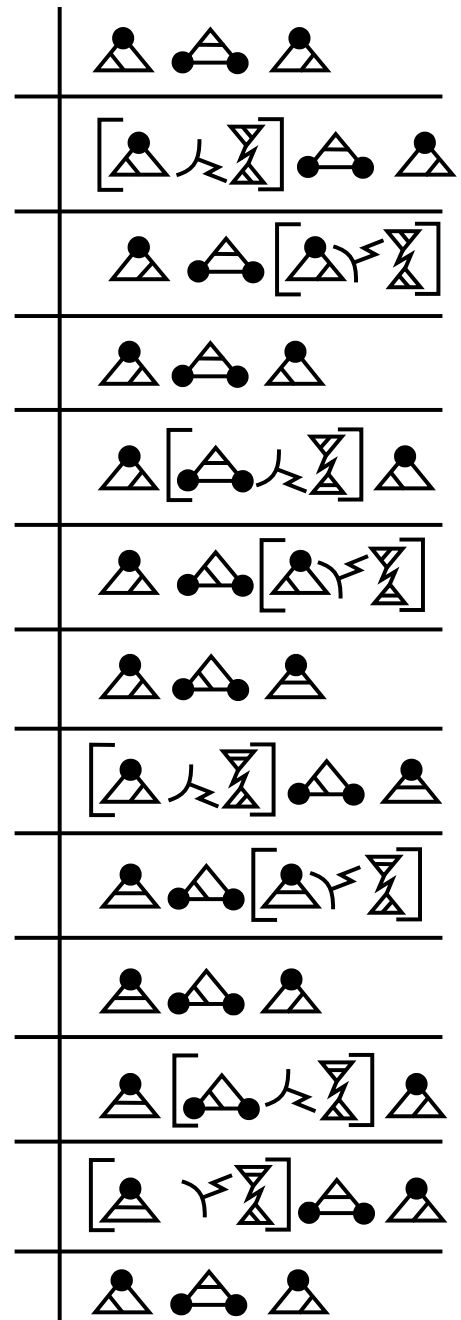
Chromodynamics may seem very foreign to some. This is probably just because for everything "above quark scales" color is always neutral. The strong force, like the weak force, doesn't ever "appear on our scale" the same way electromagnetism and gravity do, so we have no phenomena in daily life to refer to. Much the way you don't realize every pixel on screen in a grayscale image are actually composed of multiple pixels of different colors.

Mesons

Mesons are particles composed of one quark and one anti-quark held together by the strong force.

The proton and neutron, common baryons, have 3 quarks, each with half integer spin, so they also have half integer spin (2 halves cancel out leaving 1/2) which makes them fermions (in simple terms they are "matter")

The Pion, a common meson, has 2 quarks canceling out each other's spin so their spin is 0 (in simple terms they are "force carriers" aka bosons)

Baryons all have an even number of quarks.
Mesons all have an odd number of quarks
Baryons and mesons are both types of hadrons. A hadron is any particle composed of quarks.

The strong force not only holds quarks together to form hadrons, it also holds protons and neutrons together to form atomic nuclei. The easiest way to describe this is:
> while the protons and neutrons are experiencing the gluon interaction inside themselves which hold them together in a "cyclical exchange of color charges", at certain points in the cycle the gluons turn into "virtual quarks" which can be exchanged between the hadrons.

If this doesn't make a lot of sense don't worry, it is not necessary to understand the details of the strong force for Uscript, we will generalize it all as "one force" for most practical applications.

Just so we do have it defined properly, on the right you see definitions of common mesons and a simplified example of how they facilitate the strong interaction in atomic nuclei (aka the nuclear force or the residual strong force).

Image left : 4 simple examples of how to combine a quark/anti-quark into a single symbol.

Image right : A simplified time-line of a proton and neutron exchanging pions (a type of messon)
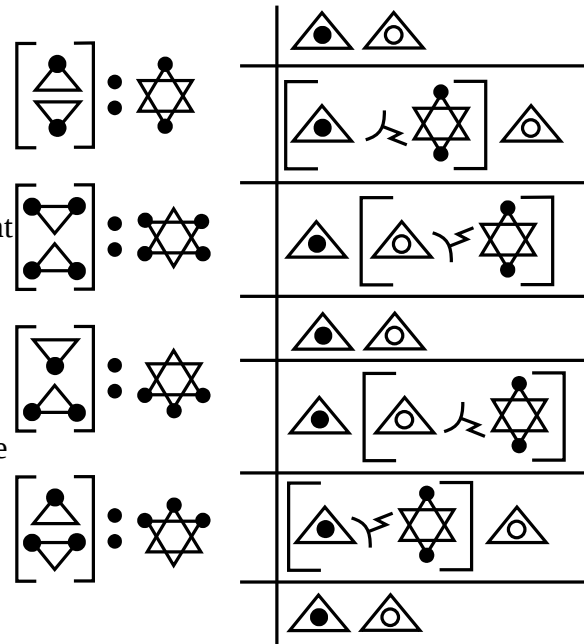
*It is technically not "correct" to think of theUp/anti-Up and Down/anti-Down as separate particles, we could define that now, but that's a much deeper conversation for another day, I'm keeping it as light as possible for now.

**to fully understand this you also need to consider color charges, all the quarks in the proton, neutron and pion have color charge, and they are cycling during this interaction, but again, lets not overdo it, we are trying to build a language, not drown readers in complex equations and physics.

***PLEASE don't ask me questions like "Hey, you used the star of David, does that imply you.......". I did NOT "use the star of David", the "star of David" is a simple geometric design. Any symbols that resemble religious, mystical, political, historical, or any other symbols used by people/groups are purely coincidental, and trivial coincidences at that. It's impossible to design simple, low complexity, elegant symbols without overlapping such things, those groups chose those symbols BECAUSE of their simple fundamental geometrical nature and/or ease of carving, writing with pen or compass/straight edge. 2 triangles overlapped like this is a "natural instinct of symmetry". None of the symbols in Uscript have any relation to ideologies, groups, organizations, etc.. they are just composite lines, dots, and geometry.

Bonds

We are going to use the triple point of water to define temperature, so we need molecules. Molecules are just bonded atoms, so it's time to clearly define and use the bond symbol, until now we have been simply grouping components into brackets to refer to composite particles.
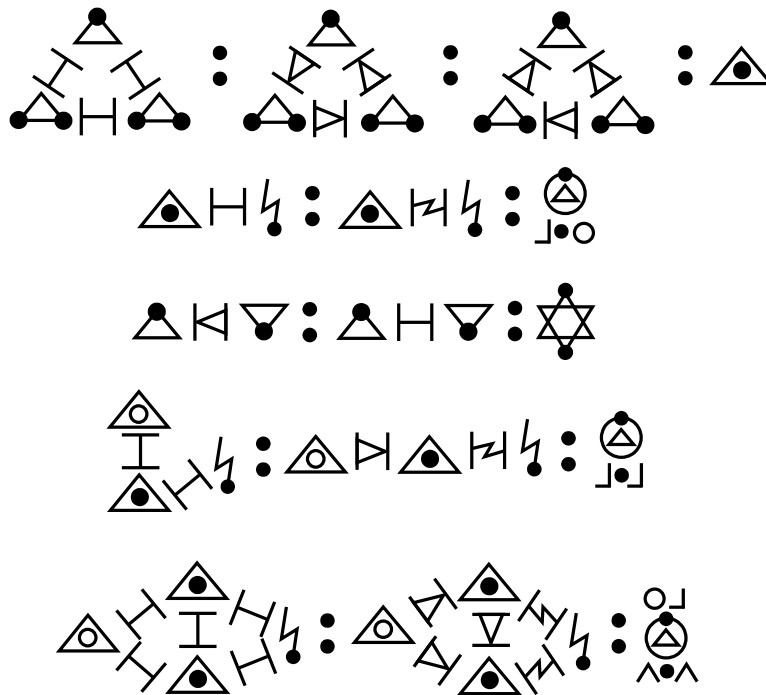
Above we clearly define the bond symbol as being a generic "binding force" as well as establish specific bond symbols for the strong force and the EM force. We can easily extend this later to create a gravity bond once we get to talking about astronomy.

Line 1: quarks bonded into a proton, there are 2 example for "strong bond" to clarify the triangle does not denote a directional element.

Line 2: proton bonded to an electron to form protium, and establish the EM bond symbol.

Line 3 : down quark bonded to an up quark to form a pion

Line 4: bonded proton, neutron, and electron to form deuterium.

Line 5: 2 protons, 2 neutrons, and 1 electron bonded to form a Helium-3 +1 cation.

As you can see there are limits to 2D representation, it would not be feasible to draw all the bonds of a large atom, especially since all the protons attract all the electrons and all hadrons in the nucleus are effectively bonded to each other in a "soupy mess".

Also notice that we are not distinguishing the strong force between quarks and the strong force between hadrons(for example pions vs gluons). If desired it would be easy enough to define 2 different strong force bond symbols, but it just seems a bit excessive at the moment.

Molecules

Molecules are easy now that we have bonds, just connect atoms with bonds. I will not use the EM bond because for now the mechanism holding atoms together is not really the "atoms attracting each other via EM force" , rather it has to do with electron orbitals and wave functions. We could perhaps make a symbol for "orbit bond", but I will leave this out for now. Details about electrons shells, or the more advanced 3D quantum wave functions can be left for "extended information databases".

Just to make sure that this is clear for any reader anywhere in the universe, we will use examples of some of the most common chemical compounds in the universe. To the the best of my knowledge these should be some common molecules found in any region of space.
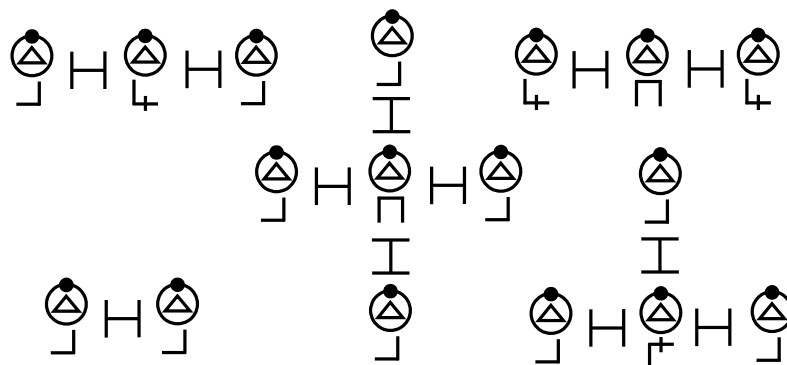
Above are 5 very simple, fundamental and common compounds.

      Top left : H O H – water
      Top right : 0 C 0 – carbon dioxide
      Middle : A cross with carbon in the middle and 4 hydrogens bonded around it – methane
      Bottom left : H H – dihydrogen
      Bottom right : Nitrogen bonded to 3 hydrogens – ammonia

Seeing as water is very common and we will use its triple point to define our temperature units, it deserves its own symbol for abbreviated usage.
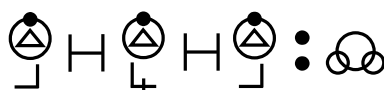
Any molecule that needs to be referenced often can easily get its own symbol assigned in the same way.

If a more efficient way of describing complex molecules is required, I would just recommend that Dscript Chemistry Notation(Dchem) is imported.

Dchem Basic : http://dscript.org/chem.pdf
Dchem Advanced : http://dscript.org/chem2.pdf

To import Dchem just provide a few example where you define a molecule in Uscript, then equate it with its Dchem equivalent, and perhaps first define the Dchem symbols for atoms.

Because Dchem can conflict with Uscript, it would be best to establish some kind of special bracket or frame to indicate that a certain writing space is using Dchem molecular notation.

States of Matter

The last thing needed to define temperature units is the states of matter. We shall stick with the common 4, solid, liquid, gas, and plasma.

There are more non-conventional states, mostly due to the strict definition we have established for solid. Glass for example is a non-conventional solid, this has to do with thermal equilibrium ground states, but we shall not get into this seeing as we wont be defining them as such yet.

On the right we define the 4 states of matter and a general symbol for cation(an ion missing electrons).

At the top 3 examples of cations shown as subsets of a symbol which will be used for cation. Below there are Four squares (remember square and circle both represent "space")

In each square at the top left you will see a gravity symbol with its force arrow pointing down. This identifies that there is a gravitational force being experienced in this space. Gravity is not necessary for states of matter, but it will help facilitate our definitions. Top to bottom :

SOLID :   A crystal-like structure stands with its peak resisting gravity, bonds are drawn between each atom to signify that they are bonded to their neighbors and the structure as a whole is a solid structure.

LIQUID: The atoms are densely packed at the bottom of the space, just as a liquid would behave because liquids do not compress but the atoms/molecules move around freely.

GAS: The atoms are distributed randomly and expand to fill the space like a gas, I did try to make the density at the bottom slightly higher than at the top, just so it can't be interpreted as "gases don't feel gravity"

PLASMA: Cations and electrons mixed.
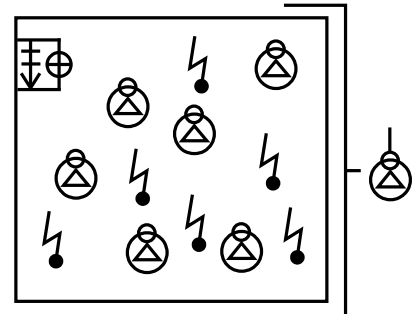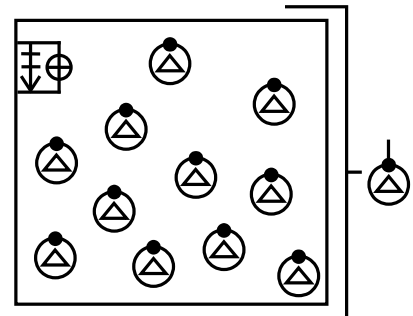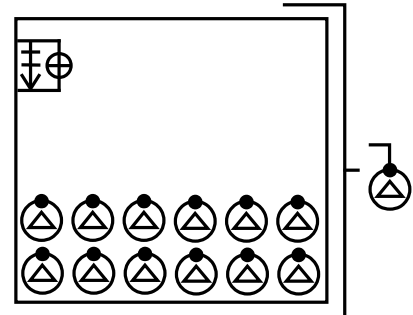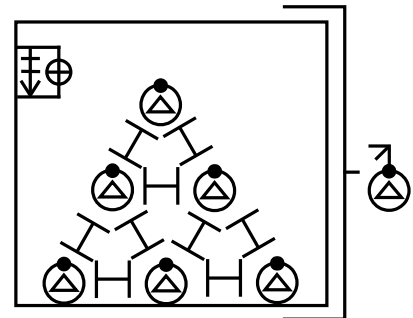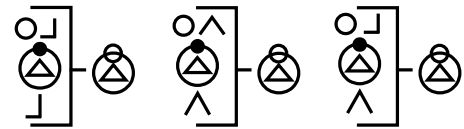
Our definition will be a purely visual intuitive one, the details of attempts to strictly categorize the "grey areas" between these states is not required for our general purposes. It can of course be included in an "extended information database".

(3D+Atom) = solid (solids hold 3d shapes)
(2D+Atom) = liquid (surface tension produces a 2d surface)
(1D+Atom) = gas (just extending the pattern)
(1D+Cation) = plasma (gas but with cations.. and electrons of course)
*We used "subset" because our examples only use atoms, thereby not excluding molecules from the states of matter. Applying the symbol to molecules is valid, and a single application to molecules clarifies that molecules can also have these states.

Temperature & Pressure

Well, it was a long road to get here, but we can finally define our final base unit... temperature.

Temperature will be defined via the triple point of water, which requires pressure be defined first. Pressure is easy to define its just force / area, but there is a small detail, specifically it's the normal force / area. We can probably get away with a visual definition that implies normal force with arrow directions, but the normal force is a very common concept in physics and it's not hard to define so it will be added here.
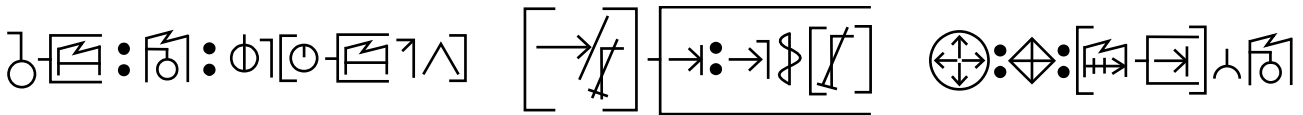
Above there are 3 statements

      Left : 2D subset(distance unit) = area unit = pi * (radius subset(distance unit) ^ 2)

      Middle : (force into angled line and and a right angle triangle with hypotenuses parallel to the line, and one side parallel to the force with the opposite angle marked) subset(normal force = force * cos(marked angle))

      Right : 4 arrows pointing outwards inside a space circle = pressure symbol = (force subset(normal force) / 2D area)

The pressure symbol is a composite of 4 outward pointing arrows. The formula should clarify the meaning and define our unit, as well ass how to produce units for area, volume, etc…

Once again we need to do a bit of unit conversion

1 Pascal = 1 Newton / meters ^ 2   (force / area)
1 Pascal = 2577968421466 Uscript Force Units / 4.7379635940184 Uscript distance units ^ 2
1 Pascal = 2577968421466 Uscript Force Units / 22.4482990182437 Uscript area units
1 Pascal = 1.148402566880853E+11 Uscript pressure units

The triple point of water is at 611.657 Pascal
611.657 * 1.148402566880853E+11 =7.024284688506419734889286E+13
7.02428468E+13 = 0x3FE2AFEE7880 ≈ 0x3.FE2*(0x10^0xB)

Ok, now to set a value for the triple point of water. Kelvin is defined as 273.16, but this was done just to make the Kelvin unit have the same magnitude as Celsius unit.

I see no reason not to just call it 1.

Temperatures below the water triple point are less than one and temperatures above are grater than 1

Sure, this makes most temperatures we encounter in our daily life less than one. We like systems that make temperatures we care about easy to handle, our temperature scales are all designed so that 0 is "cold" and 100 is "hot", they are designed for the range near the human body temperature. Human body temperature is human centric and arbitrary in the grand scheme of the universe., so we will use 1 for the triple point of water instead.

On the right we define our temperature units

The graph is labeled as :

Vertical axis temperature (increasing downwards)
Horizontal axis pressure (increases left to right)

The graph is divided into three sections
     -top section : water is solid
     -lower left section : water is gas
     -lower right section : water is liquid

The dot represents the triple point.

The triple point has its 2 axis values labeled

The labels are a subset symbol with extended lines coming close enough to the point to identify that they refer to the value of the point. The subset symbol an be written as "Dot Subset(Value)" or "Value Superset(Dot)"
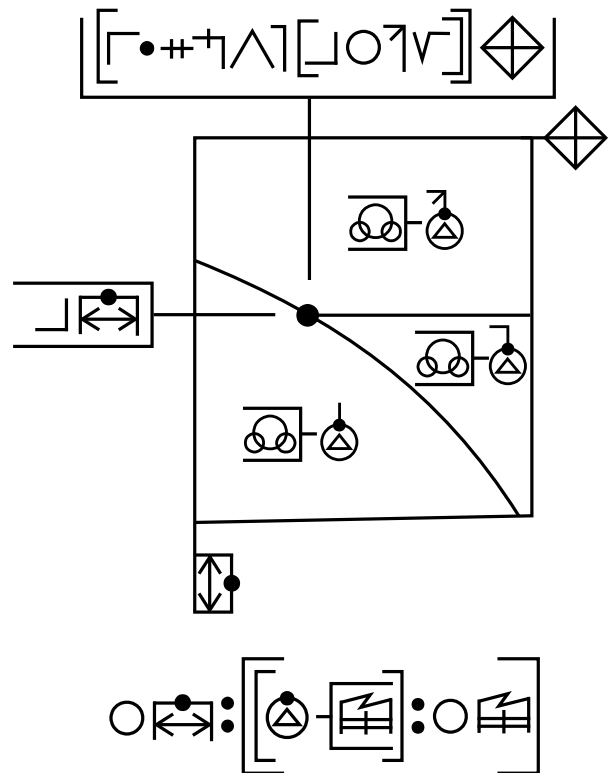
Vertical label at top : (0x3.FE2 *(0x10 ^ 0xB)) pressure units

Horizontal label on left : 1 temperature unit

Below the graph is the the statement
     0 temperature units = ((atom subset(velocity))= 0 temperature)

The graph establishes our temperature units with 1 being the triple point of water
The lower statement establishes 0 temperature units as being absolute zero

We are using a kinetic interpretation of temperature as it is the simplest. There are many more interpretations, but this is the easiest and most likely to be the most universally obvious without resorting to complex equations or quantum mechanics.

We could further clarify our temperature scale by defining the temperature values of the triple points of more compounds, but one is enough I think. Sso far I have only used linear scales, we have not done anything anywhere in our document to imply anything else, such as log scale units. The graph section boundaries may not be perfect but that is a bit trivial at this point because we are only discussing the triple point itself.

It would also be easy to go an extra step and define a new symbol for "triple point" via various methods
     eg. triple point = (compound = solid AND liquid AND gas)
       or (x compound subset(triple point)) subset (temp=x, pressure=y)

…. Et voila!… a full compliment of base units. We can now derive any units we want.

Uscript Astronomy

Astronomical terms will need to be defined, they are good for discussing the universe we live in and great linguistic tools as well. We can't just define stars as "shiny twinkle things in the night sky" because we can't assume the reader has the experiences of "sky"," shiny", or "night", so physical properties and mechanisms will be how we define astronomical terms.

Stars & Fusion
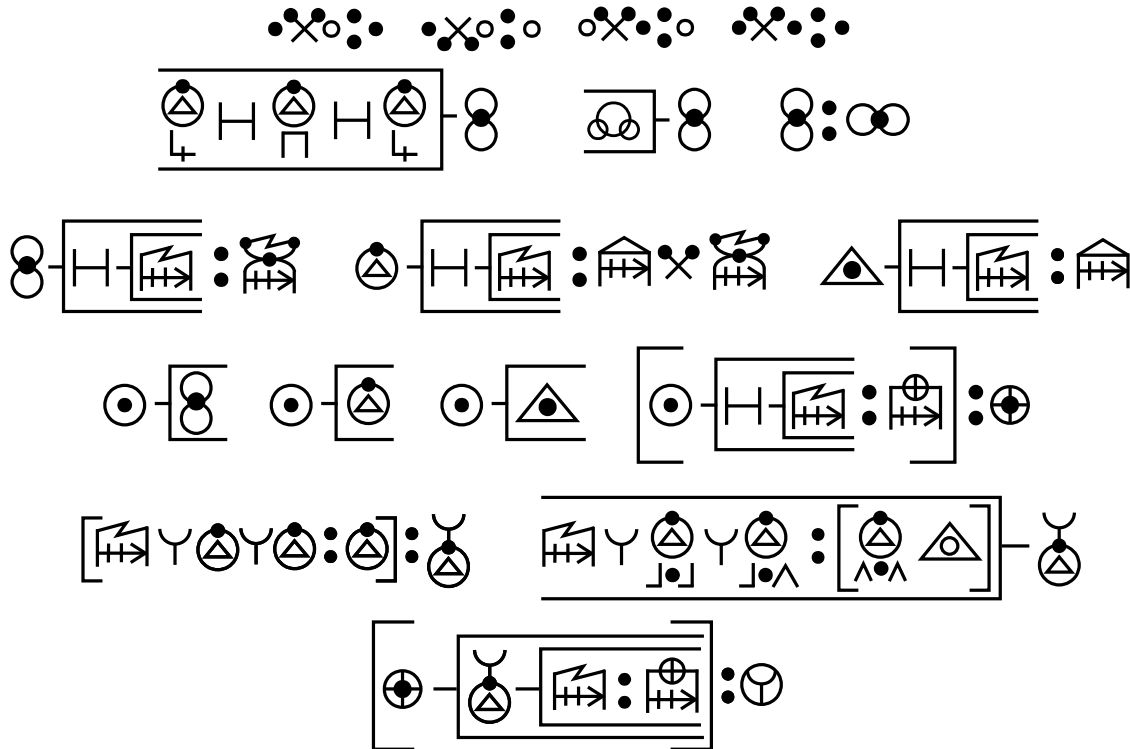
First line   We clarify that the top row of the truth table is the first variable and the second row is the second variable. This way we can create FIRST-GATE and SECOND-GATES gates. Eg FIRST-GATE means "require first variable, second variable optional",

Second line   (C02)subet of molecule symbol   water subset of molecule symbol
              Also 2 versions of molecule symbols. For ease of use it can be rotated 90 degrees

Third line   molecule subset(bond subset(force)) = electromagnetism
              atom subset(bond subset(force)) = strong force FIRST-GATE electromagnetism
              proton subset(bond subset(force)) = strong force

Fourth line   *new symbol "compound particle", will use for things like "system" and "body"
              system subset(molecule)   system subset(atom)   system subset(proton)
              (system subset(bond subset(force)) = gravity) = "astronomical body" system
                      *any system held together by gravity is an astronomical body in Uscript
Fifth line   (force + atom + atom = atom) = fusion
              (force + deuterium + tritium= (helium-4 neutron)) is a subset of fusion

Sixth line   (astronomical body subset(fusion subset(force = gravity)) = star symbol

This has clearly defined stars: a gravity bound body fusing atoms with gravitational force is a star.
        *technically one could argue black hole fuses atoms, and I suppose they do, but our definition of fusion states that the product of fusion is an atom, which eliminates black holes as far as I know. We will define black holes separately so this is not a problem, they have plenty of unique identifying properties. But still, technically if you can find examples of black holes using gravity for fusion, then Uscript says they are stars.

Planets, Moons, Asteroids, etc…

We could define "all astronomical bodies with 0 gravitation induced fusion" as planets / meteors / asteroids / etc.. but that would not be a very useful category. To be honest asteroids vs. meteors vs. comet are hardly that relevant universal distinctions.

Instead we will create several categories based on 3 qualities.
1. Spherical – being large enough for gravity to force the object into a spherical shape (this criteria is currently on of the prerequisites for our current definition of planet.
2. Orbiting a star – Orbiting a star
3. Orbiting a non-star

This will create 8 major (as well as some parent categories)

Spherical Stellar Orbiters   Non-spherical Stellar Orbiters
Spherical Free Bodies   Non-spherical Free Bodies

Spherical Orbiters of Stellar Orbiters   Non-spherical Orbiters of Stellar Orbiters
Spherical Orbiter of Free Body   Non-spherical Orbiter of Free Body

Spherical Stellar Orbiter is basically a planet.
Spherical/Non-spherical Orbiters of Stellar Orbiters are basically moons.
Oumuamua is a Non-spherical Free Body

Some of the other categories are not very commonly used, if at all (eg. it's not often we discuss non-spherical moons of rogue planets) but this gives us clear categories that will serve our purposes, and who knows, maybe there is life on rogue planets traveling through intergalactic space who talk a lot of "Free Bodies" and their orbiters. The term "spherical" will require some more math concepts and functions.
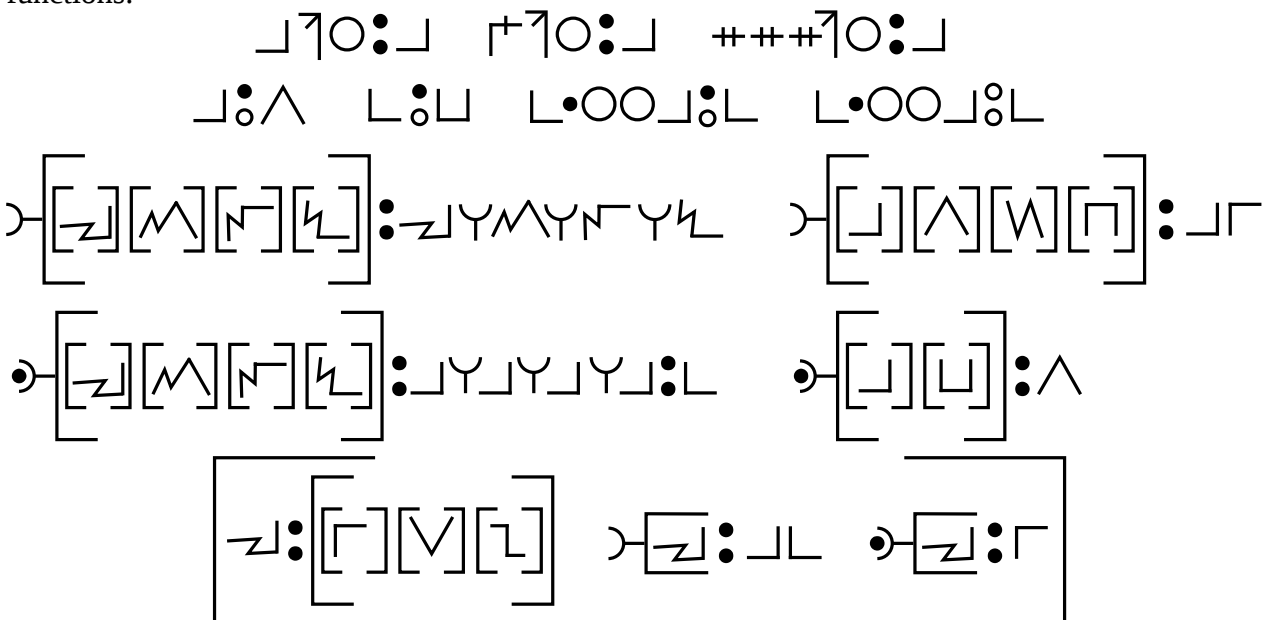
Top line : 1 ∧ 0 = 1   7 ∧ 0 = 1   0xFFF ∧ 0x0 = 1
Second line : 1≠2   4≠9   0x4.001≠4   0x4.001≈4
Third line : Sum(a,b,c,d) = a+b+c+d   Sum(1,2,10,6) = 0x13 (dec 19)
Fourth line : Count(a,b,c,d) = 1+1+1+1 = 4   Count(1,9)=2
Fifth Line : (a=(3,5,12)   Sum(a)=20   Count(a)=3)

This now gives us clarification that "anything to the power of zero" equals 1, a new symbol for "not equal to", the ability to use arrays, a Sum-Array function, and a Count-Array-Elements function. Just a few more definitions and we can use expressions to define spherical.

   *I have intentionally avoided drawing a sphere wire-frame. I do not want to go down the road of drawing complex pictures. I want to keep everything as simple and 2D as possible. We have used a few graphs, but thats as far as I want to go. I only made the exception at the very beginning where I used a cube and tesseract wire-frame to define our X-dimensional radical.
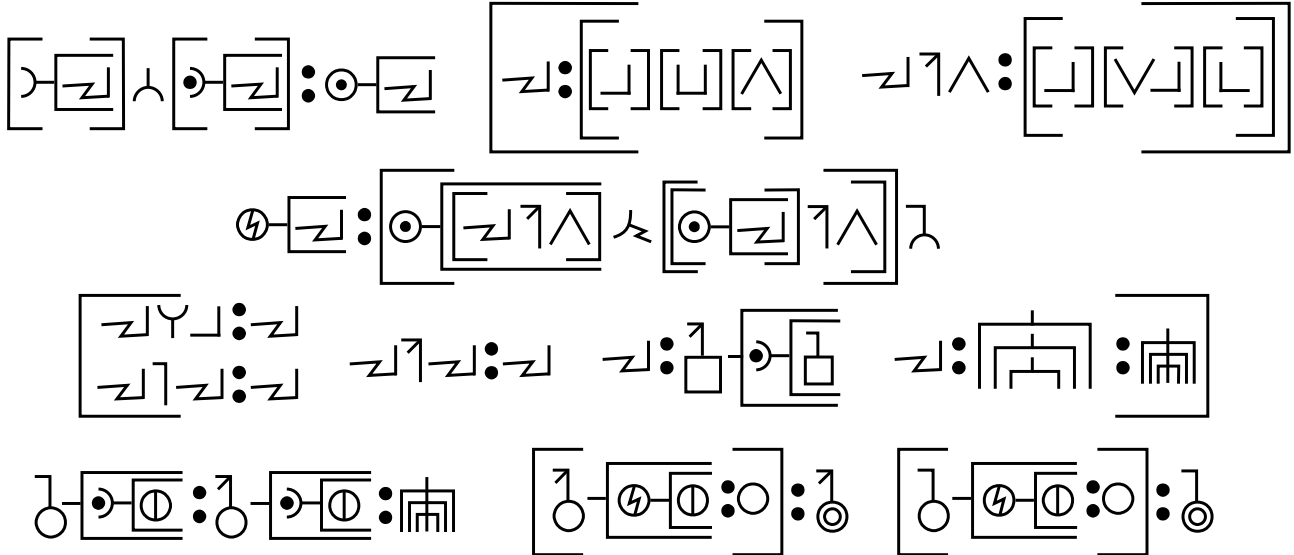
Top Line :   Sum(a) / Count(a) = Mean(a)   (a=[1,9,2]   a^2=[1,81,4])
Second line : Standard-Deviation(a) = (Mean(a^2) - (Mean(a)^2)) Square root
Third Line :
      a+a=a   a^a=a   a=3D-Cubic Sub(Count(2D Space))   a=Infinity symbol =Infinity symbol
      a*a=a
Fourth line :   2D-Circle Sub(Count(diameter))=3D Sphere Sub(Count(diameter)) = infinity
              (3D sphere Sub(Standard Deviation(Diameter)=0)) = Perfect Sphere
              (3D circle Sub(Standard Deviation(Diameter)=0)) = Perfect Circle

Here we have defined some basic statistics functions and and defined the concept of infinity. Using these 2 terms we have essentially said "There are infinite 2D sub-spaces in a 3D space", "Circles and spheres have an infinite number of diameters to measure", and "If all the diameters are the same it is a perfect Sphere/circle"

*Technically when we defined circle and sphere we implied "perfect" with the area and volume equations, I will probably just change those definitions to be "approx" instead of equals.

*I thought long and hard about trying to make a simple and elegant symbol for infinity. I do not think our current one is intuitive, and it can possibly be mistaken for 2 circles if drawn to small or messy. In the end I landed on this one because it can at least be argued to be a combination of radicals that have a relatively intuitive meaning that could be implied to mean infinity. The parent of a subset of a subset of a subset could be extended to mean "parent of everything", we will not use it like that though, just as the mathematical term infinity. I also made it 3 levels because I think I may want to use the second tier subset concept later, symbols for arbitrary levels will be done with numbers though. Akin to the way English has parent, grandparent, and then just add "great great great"

*I went straight to standard deviation and skipped variance, the same way we use sigma, just square it to get variance.
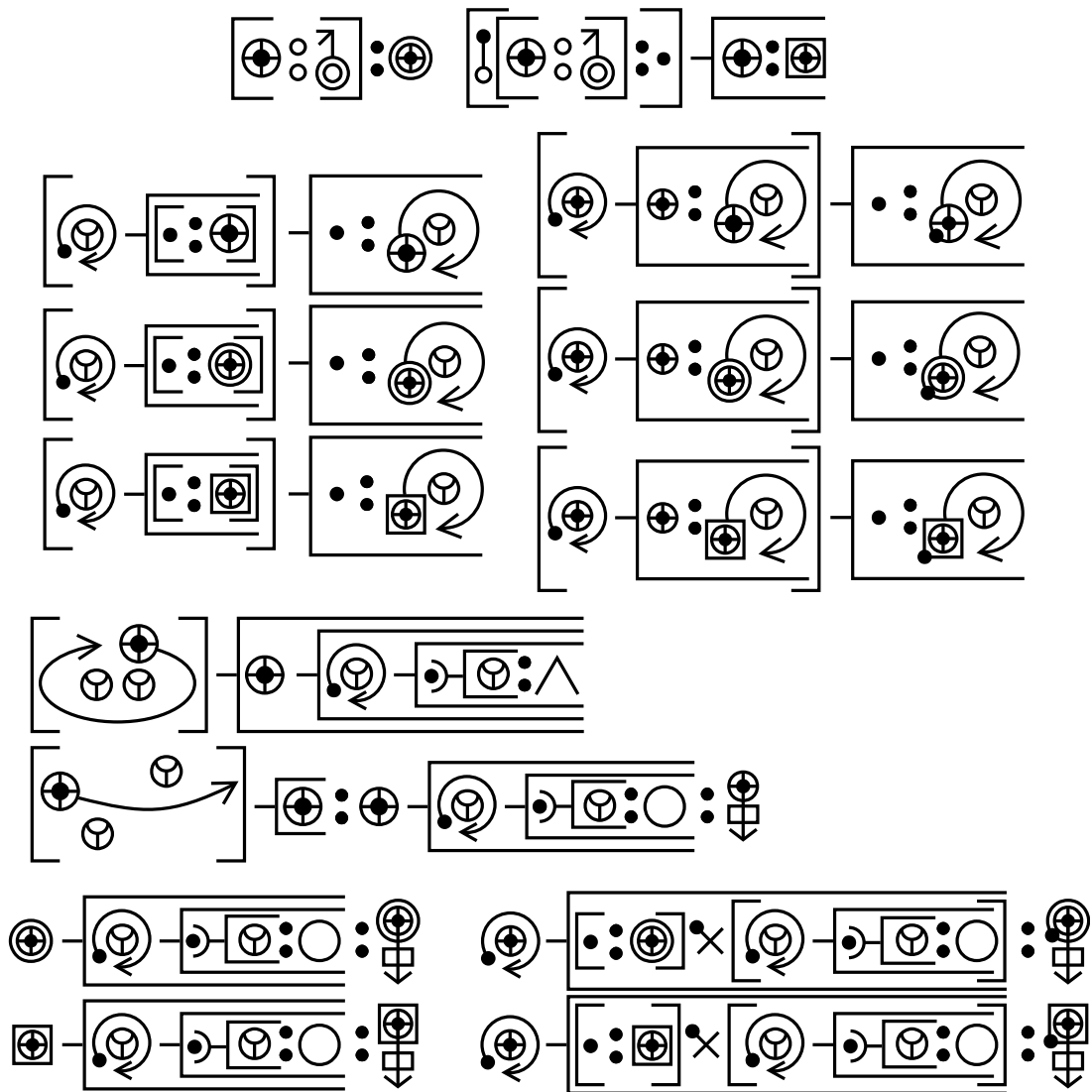
astro-body = A-body = astronomical body   S-Orbiter =Star-Orbiting

P = particle   Fbody=Free body=Astrobody not orbiting any stars   non-speherical = Nsphere

Top line: (astronomical body ≈ perfect sphere) = spherical astronomical body
                   (invert(astro-body ≈ perfect sphere)=TRUE) sub(astro-body=non spherical astro body)

Upper Left   (particle-orbit-star sub(particle=astro-body))sub(particle = star-orbiting-astro-body)
                        =astro bodies that orbit stars
                   (P-orbit-star sub(particle=sphere-astro-body))sub(P = star-orbiting-sphere-astro-body)
                        =spherical astro bodies that orbit stars (planets)
                   (P-orbit-star sub(P=Nsphere-A-body))sub(particle = S-Orbiter-Nsphere-A-body)
                        =non-spherical bodies that orbit stars (asteroids etc..)

Upper Right   (P-orbit-A-Body sub(A-body=S-Orbiter-A-Body))sub(particle = S-Obriter-Abody-Orbiter)
                        =astro bodies thats orbit astro bodies that orbit stars = moons / astro body satellites
                   (P-orbit-A-Body sub(A-body=S-Orbiter-sphere-A-Body))sub(particle = S-Obriter-sphere-Abody-Orbiter)
                        =spherical astro bodies thats orbit astro bodies that orbit stars = spherical moons / satellites
                   (P-orbit-A-Body sub(A-body=S-Orbiter-Nsphere-A-Body))sub(particle = S-Obriter-Nsphere-Abody-Orbiter)
                        =non-spherical astro bodies thats orbit astro bodies that orbit stars = non-spherical moons / satellites

Lower 4 lines   (A-body orbiting 2 stars)sub(A-body sub(P-orbit-star sub(count(star)=2)))
                   (A-body passing but not trapped by stars) sub(A-body) = A-body sub(P-orbit-star sub(count(star)=0)=Free body
left line 3   sphere-A-body sub(P-orbit-star sub(count(star)=0)) = spherical free astro body   =rogue planet
left line 4   Nsphere-A-body sub(P-orbit-star sub(count(star)=0)) = non-spherical free astro body
right line 3P-orbit-A-body sub((P=sphere-A-body)AND(P-orbit-star sub(count(star)=0)) = sphere moons/satellites of Free bodies
right line 4P-orbit-A-body sub((P=NsphereA-body)AND(P-orbit-star sub(count(star)=0)) = Nsphere moons/satellites of Free bodies

Galaxies

Galaxies are a bit arbitrary, but still obvious observable astronomical bodies.

Galaxies are clusters of stars bound by gravity, but this becomes hard to define clearly. How many stars constitute a galaxy? surely 2 stars in a binary star system do not count. 100? 1000? 10000? etc.. I have seen it defined as "millions or billions of stars" which is still arbitrary, 999,999 is not a galaxy?

We are not sure, we think nearly all galaxies have a super massive black hole at their center, but it seems that according to models it is not required that there be super massive black holes at the center of all galaxies. So we cant use that as part of the definition. So the Uscript definitions will differ from our common definition somewhat.
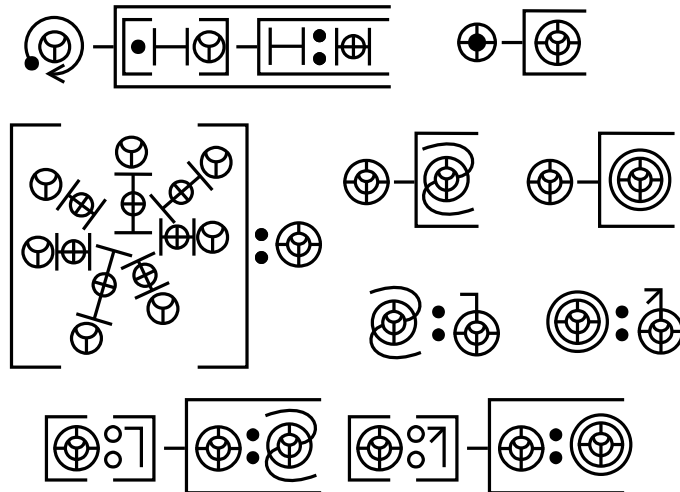
Top line : particle orbiting star sub(particle bond star) sub(bond=gravity bond)
     *"the bond between a solar satellite and and the star is a gravity bond"
    astronomical body sub (galaxy symbol)
     *a galaxy is a subset of astronomical bodies

Mid section:
  left : (many stars gravitational bound to a common center) = galaxy symbol
  right : galaxy sub(spiral galaxy) galaxy sub(elliptical galaxy)
     spiral galaxy = 2D galaxy elliptical galaxy = 3D galaxy

Bottom line : (galaxy approx 2D) sub(galaxy = spiral galaxy)
     (galaxy approx 3D) sub(galaxy = elliptical galaxy)

Here we have defined
  A galaxy is any cluster of stars gravitationally bound to a common center.
  There are (at least) 2 sub categories of galaxy, spiral/2D and elliptical/3D

There are 2 symbols for each type, I was a bit reluctant to create double symbols for a single concept, but it does help with the definition, 1 version of the symbol graphically depicts the spiral/round shape, and the other uses their dimensionality (Spiral galaxies are flat and elliptical galaxies are 3D spheroid)

I will, for now, refrain for using "star count" in the definition, it's just so arbitrary. So in Uscript you can argue that a binary star system is a galaxy. The visual definition had an empty center so this has been left open too, all we require is an astronomical body composed of stars.

Black Holes

Black holes can be defined via the Schwarzschild radius.

Schwarzschild radius = (2 * G * Mass) / speed of light ^ 2

Uscript speed units are by definition the speed of light due to our base units being based on a photon (distance a photon travels / time it take it to travel). This make the equations surprisingly simplified. (in other words our speed unit is a natural unit)

Schwarzschild radius = 2 * Uscript-G * Mass / 1 ^2

Schwarzschild radius = 2 * Uscript-G * Mass

We can eliminate the speed of light from the equation because the speed of light in Uscript is 1.

Lets try this with the super massive black hole in the Andromeda galaxy to test it out

SMBH-Andromeda

Mass = 3.4*(10^38) Kg

Schwarzschild radius = 5*(10^11) Meters

Mass Uscript = 3.4*(10^38) / 9.109*(10^-31) Kg = 3.7325*(10^68)

2 * (3.203*(10^-57)) * (3.7325*(10^68)) = 2.391 * (10^12) Uscript distance units

2.391 * (10^12) * 0.211 = 5.045*(10^11)

Yup, it works. Now lets define a black hole

Top line :   1 superset(2)   1 superset(3)   1 superset(F)   E superset(F)

2 subset(1)   6 subset(7)

*this should be sufficient for "a smaller number is a subset of a larger number and a larger number is a superset of a smaller number" allowing use to use subset superset as less-than and greater-than

Second line :   (a superset(b)) = (b subset(a)) = a superset b = b subset a

*this creates a simpler way to use the symbol. I have retroactively used this in the definitions for isotopes and ions, so this definition should be moved up into the early math section of the key. I may deprecate this usage later though.

Third line :   Spherical astro body subset (diameter < 4 * Mass * Uscript-G) = Black hole symbol

*this defines a black hole as a type of spherical astro body with a diameter smaller than 2*Schwarzschild radius(they appear and are spherical for all measurable and observable intents and purposes). I tried very hard to come up with a good symbol, but nothing worked, so I used one that is essentially a combination of 4 inward pointing arrows, akin to how we made the pressure symbol. I will not define the symbol as such because all astro bodies have gravity. I didn't want to start referencing singularities or other approaches, so we can just consider it arbitrary symbol.

Fourth line :   ((photon distance blackhole)<blackhole sub(Schwarzschild radius))=blakhole absorb photon

*in other words "a photon within the Schwarzschild radius is absorbed by the black hole.". This is just for extra clarity of definition and not necessary.

Neutron Stars

We may call them stars, but the truth is neutron stars don't really fall into the Uscript definitions of star. Neutron stars are more like giant atoms. We define stars in Uscript as "astronomical bodies that fuse atoms with gravity", and you could say that during the formation of a neutron star all the atoms are fused into one giant atom, but once it is formed there isn't really much fusion going on to speak of. I suppose you can argue that "all atoms falling into a neutron star will fuse with it, so there is fusion always going on".

Our current model of neutron stars also expects there to be a shell of ionized atoms on the surface, which one would imagine can fuse. But this is much different from the fusion going on at the core of "normal stars", and since most of this "fusion" is fusing with the star itself, its a real stretch to argue this. In Uscript we won't bother with this debate, we will avoid, the definition of whether a neutron star is a subset of star, and define it directly as a subset of astronomical bodies. I leave classifying it as a subset or not a subset of "Uscript star" open to debate for now.

Defining a neutron star is surprisingly simple. While they can vary slightly from one to another, they have a limited mass range, and limited ratio of neutron to proton composition (yes, we expect there to be a certain non-zero amount of protons and electrons).
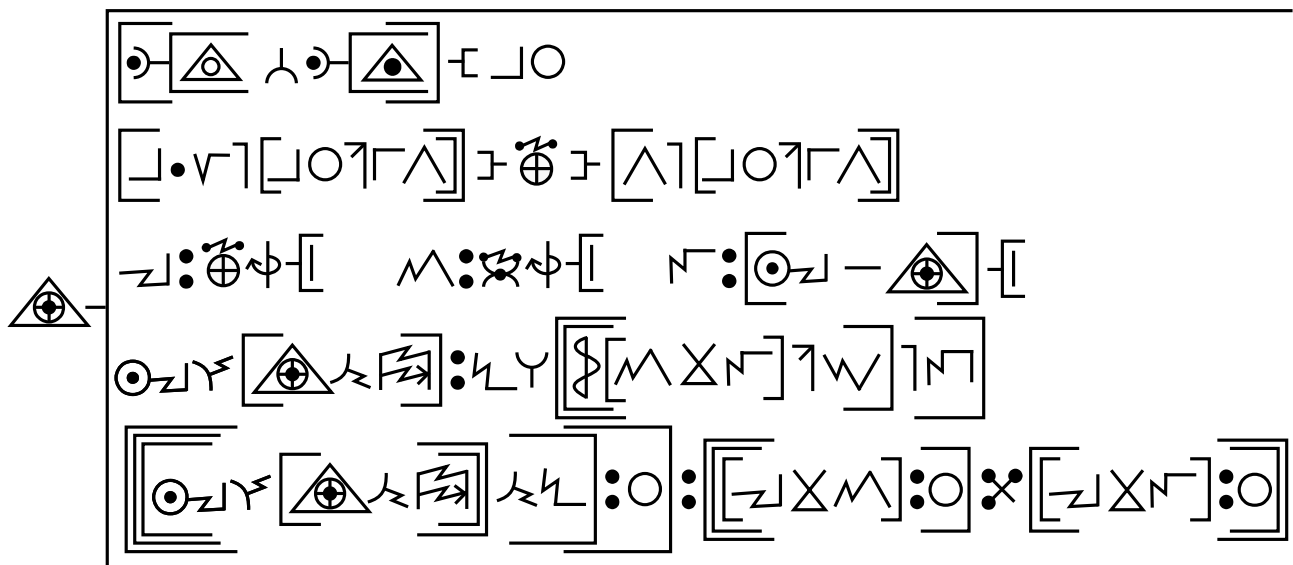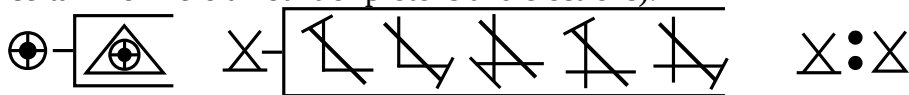
Top line : Astro body superset(neutron star)   angle superset (various angles)   angle=angle
*Here we create a symbol for neutron star and a general symbol for angle using some angles we defined previously
Large subset of Neutron star :
     First line : (count(neutron) / count(proton)) > 16   *more than 16 neutrons per proton
     Second line : (0x1B*(0x10^0x32)) < mass < (0x5*(0x10^0x32))
                *mass between 1.2 solar mass and 3.5 solar mass (roughly)
     Third line: a=mass spin sub(axis) b=charge spin sub(axis) c=(system-a line neutron star) sub(line)
     Fourth line : system-a absorb (neutron star radiate energy) = d+((sin(b angle c)^e)*f)
          *by varying d,e,and f you can create any pulse amplitude by rotating the angle from magnetic pole
     Fifth line :   (((system-a absorb(neutron star radiate energy))-d)=0) =
            (((a angle b)=0) AND-OR ((a angle c)=0))

These statements basically say "A neutron star has at least 16 time more neutrons than protons, its mass is between 1.2 and 3.5 solar mass, observers can see increased brightness when aligned with the magnetic pole, and if observers only see a baseline brightness then the magnetic pole and rotational pole are aligned and/or the observer is aligned with the rotational pole."

Evaluations, Comparisons, Conditionals & Loops

Before I move on to more complex concepts like Entropy and Life we will need more tools for expressing complex logic and procedural processes. These not only enable us to describe a processes better, they also enable more mathematical expressions and greatly abbreviate some definitions that would require many more lines and graphics to express otherwise.

First and foremost we require boolean evaluations and comparisons. A bracket with a dot or circle in the bracket line indicates that this is an evaluation, not a descriptive expressions. The Dot or circle represents that we are going to reduce this expression to True or False.
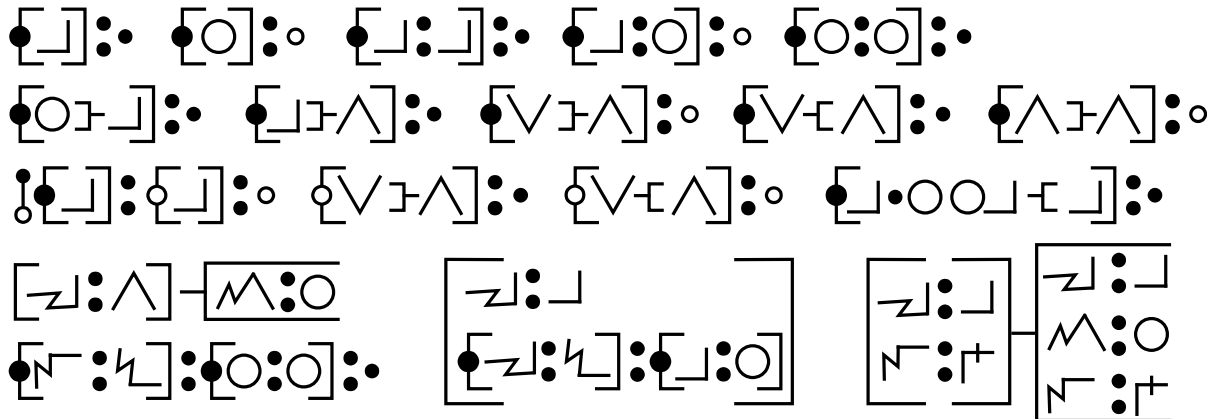
Top line :   eval(1)=TRUE   eval(0)=FALSE
              eval(1=1)=TRUE   eval(1=0)=FALSE   eval(0=0)=TRUE

Second line : eval (0<1)=TRUE   eval(1<2)=TRUE
              eval(5<2)=FALSE   eval(5>2)=TRUE   eval(2<2)=FALSE

Third line : invert eval(1) = inv-eval(1) = FALSE   inv-eval(5<2)=TURE
              inv-eval(5>2)=FALSE   eval(0x1.001 > 1)=TRUE

Bottom :
      Left :   (a=2) sub(b=0)   eval(c=d)=eval(0=0)=TRUE
      Middle :   (a=1;   eval(a=b)=eval(1=0))
      Right :   (a=1; c=7) sub(a=1, b=0, c=7)

The first lines give use an evaluator bracket, all non zero values are true and zero if false. The equals symbols inside the evaluator bracket is an equality comparison (== in C programming).

The second line gives us greater-than and less-than comparisons

The third line give us an inverted evaluator which gives us easy access to not-equal-to, greater-than-or-equal-to, and less-than-or-equal-to (!=, >=, <= in C)

The bottom sections establishes that undefined variables are zero. (if want a NULL, that can be done, but I'm not gonna bother for now as it is not necessary and I don't need it for this doc)
Now that we have evaluations we can create conditionals
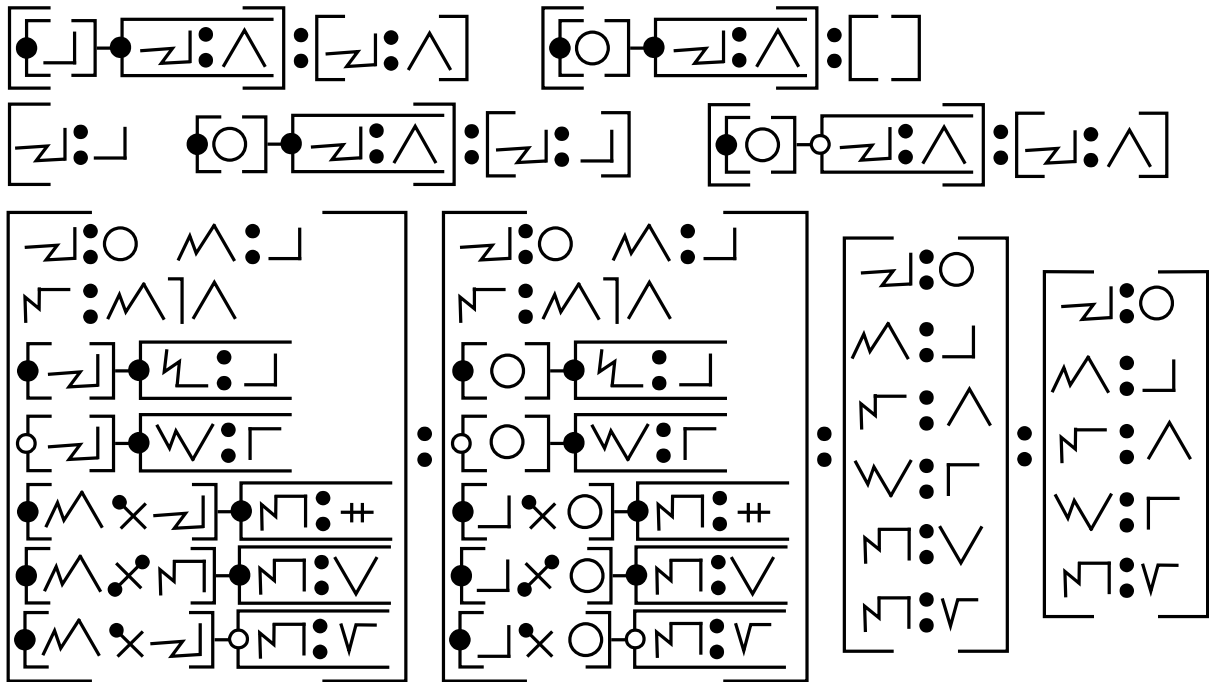
*I am using C programming format here, thats why there are extra semi-colons and other notation. Also C has no XOR so I'm using "^^" and if(!x) for inverted-eval(x).

Top line :   (if(1){a=2;}) = (a=2)   (if(0){a=2;}) = ()*empty brackets, nothing occurs

Second line : (a=1; if(0){a=2;}) = (a=1;)   (if(0){}else{a=2;}) = (a=2)

Bottom :

| a=0; b=1;		| a=0; b=1;		| a=0;		| a=0;
| c=b*2;		| c=b*2;		| b=1;		| b=1;
| if(a){d=1;}		| if(0){d=1;}		| c=2;		| c=2;
| if(!a){e=3;}	=	| if(!0){e=3;}	=	| e=3;	=	| e=3;
| if(b&&a){f=0xF;}		| if(1&&0){f=0xF;}		| f=5;		| f=0x0F;
| if(b^^e){f=5;}		| if(1^^0){f=5;}		| f=0x0F;		|
| if(b&&a){}else{f=0xF;}		| if(1&&0){}else{f=0xF;}		|		|

This gives a nice set of conditionals.

I have not defined a method for doing a full IF(a){b}ELSE{c}. this could be defined several ways, but for now we can juts make 2 statements by combining an IF with an IF-ELSE("if subset with empty circle" instead of dot at the 3 line intersections) or combining an IF with an INVERTED-EVAL-IF.

We have also defined how operations are executed in sequence, how variables are overwritten, and reinforced a few principles I felt might benefit from extra clarity. Some may feel there are potential operations and combinations that are not clear enough, I think this is more than enough to say it is "clearly defined" but it can always be expanded with more examples.

Well…. Seeing as we have this much, there is no reason not to give it full programming capability, for that all we need is loops (yes, goto would work too, even with loops adding a format for goto is arguably very useful.. but I shall leave that out for now)
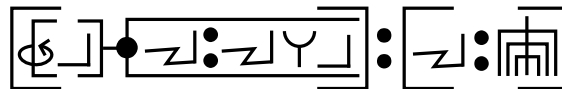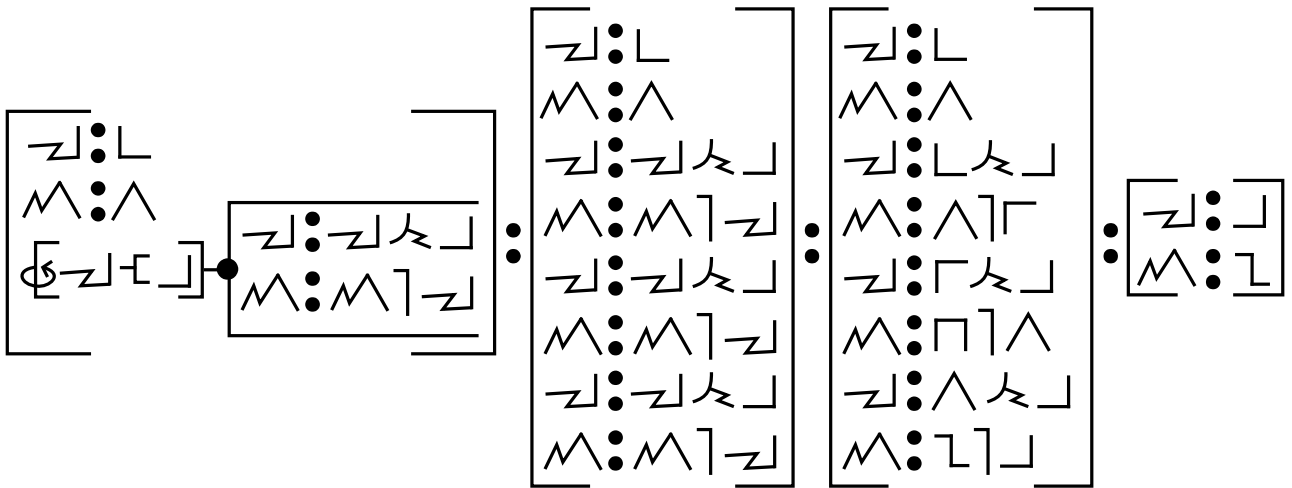
Top section :   a=4; | | a=4; | | a=4; | | a=1;
                b=2; | | b=2; | | b=2; | | b=12;
                while(a>1){ | | a=a-1; | | a=4-1; | |
                 a=a-1; | = | b=b*a; | = | b=2*3; | = |
                 b=b*a; | | a=a-1; | | a=3-1; | |
                  } | | b=b*a; | | b=6*2; | |
                              | | a=a-1; | | a=2-1; | |
                              | | b=b*a; | | b=12*1; | |

Bottom line : (while(1){a=a+1;}) = (a=infinity)

The top example should make loop programming logic perfectly clear, more examples can be added to reinforce the principles or add more optimizations, but this more than enough for now. The bottom line just reinforces the meaning of our infinity symbol. As you can see, now that we have loop logic we can discuss many more abstract concepts and math formulas.

Just for good measure I'll also add array element referencing,
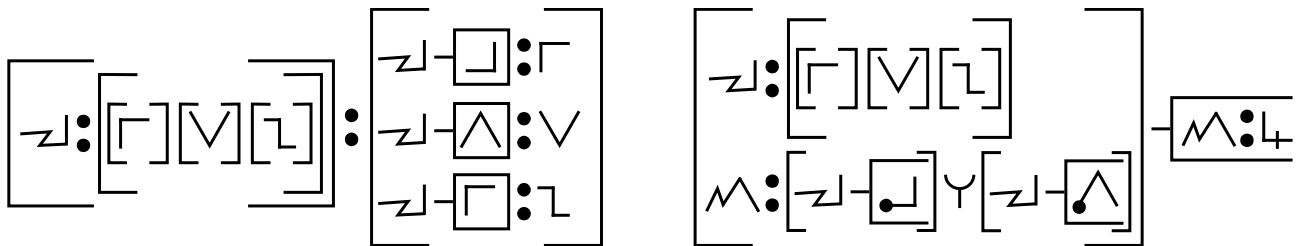
Left : (a={3,5,12};) = (a[1]=3;   a[2]=5;   a[3]=12;)
Right : (a={3,5,12};   b=(a[1]) + (a[2])) subset (b=8)

This defines how to refer to array elements.

I have not defined how to refer to particles, variables, and array elements above more than 15, there are various ways to do it which can be defined later should it be needed.

                                 **yes I know.. "Bad programmer!".. I don't start with element 0.

Entropy

The entropy axis of the time/entropy symbol has not yet been adequately defined. Time is clear, but we have not ensured that the horizontal entropy axis meaning is clearly defined. Entropy will be needed for higher level definitions, like life and intelligence, so first we must clear this up. First a few things to help us with out definition. Speci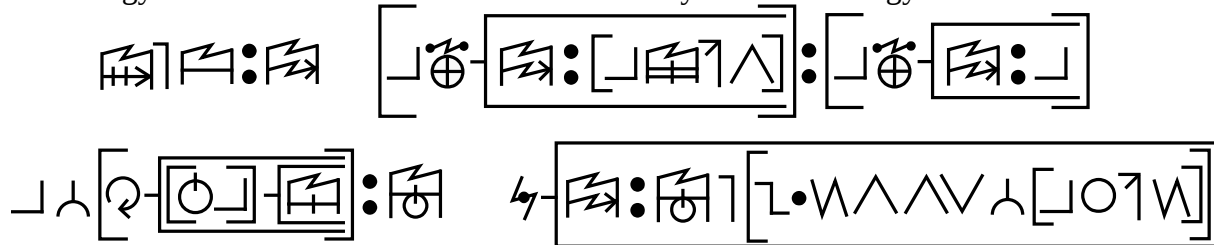fically energy units, matter energy equivalence and photon energy calculation because we want to discuss systems and energy.

Top line : force * distance = energy
  1 mass sub(energy = (1 speed of light ^ 2)) = (1 mass sub (energy=1))

Bottom line : 1/(cycle sub(1 circumference) sub(time)) = frequency
  photon sub(energy=frequency * (0xC.A225/(0x10^0xA)))

A quick test of these…
  1 Joule = 1 Newton*Meter   1kg = 1 * (3E+8m/s)^2   1kg = 9E+16 Joules
  1kg = 1.097E+30 Uscript mass units
  1 Newton = 2.57796E+12 Uscript Force Units
  1 Meter = 4.73796359 Uscript distance Units
  1 Joule = 2.57796E+12 * 4.73796359 = 1.221E+13 Uscript energy units
  mass = energy so…   1.097E+30 / 1.221E+13 = 8.9844E+16 Joules

Ok.. Now photon energy
  Planck constant (h) = 6.626E-34 Joule Seconds
  1Hz photon = 1 * 6.626E-34 Joules = 6.626E-34 Joules
  1Hz = 1/ 1.4204E+9 Uscript time units = 7.04E-10 Uscript frequency units
  7.04E-10 * Plank constant = 6.626E-34 * 1.221E+13 Uscript energy units
  7.04E-10 * Plank constant = 8.0903E-21 Uscript energy units
  Plank Constant = 8.0903E-21 / 7.04E-10 = 1.149E-11

  (Arbitrary) UV photon frequency = 7.8E+14 Hz
  UV photon energy = 6.626E-34 * 7.8E+14 = 5.1682E-19 Joules

  (Uscript 1hz in feq units * frequency)*Uscript plank constant = Uscript energy units
  (7.04E-10*7.8E+14)*1.149E-11=6.3093E-6
  6.3093E-6 Uscript energy units / 1.221E+13= 5.167E-19 Joules

  Uscript plank constant in Hex : 0xC.A225C1682C16/(0x10^0xA)

Yup. That works too. So that is the Uscript plank constant, and we can use 1 instead of c^2.

Next we define "Entropy". This section could be a source of contention, but please remember, we are not trying to define entropy exactly as we define it currently, the Uscript term we are creating is just best translated as entropy, it's not meant to be exact.

*notice I did not use e=mc^2, and instead say "mass sub(energy=c^2)". While there is an energy to mass equivalence, and you could get away with saying "all mass is made of energy", you cannot say "all energy composes mass". **debatable, I suppose, let's not get into it now.

Entropy is a tricky one to define, the majority of people know the word but have a foggy or even misinformed understanding of it. We usually define it using thermodynamics. Uscript will take a slightly different approach. Introductions to entropy usually start with some examples, and so will we, but we'll use some different types of examples.
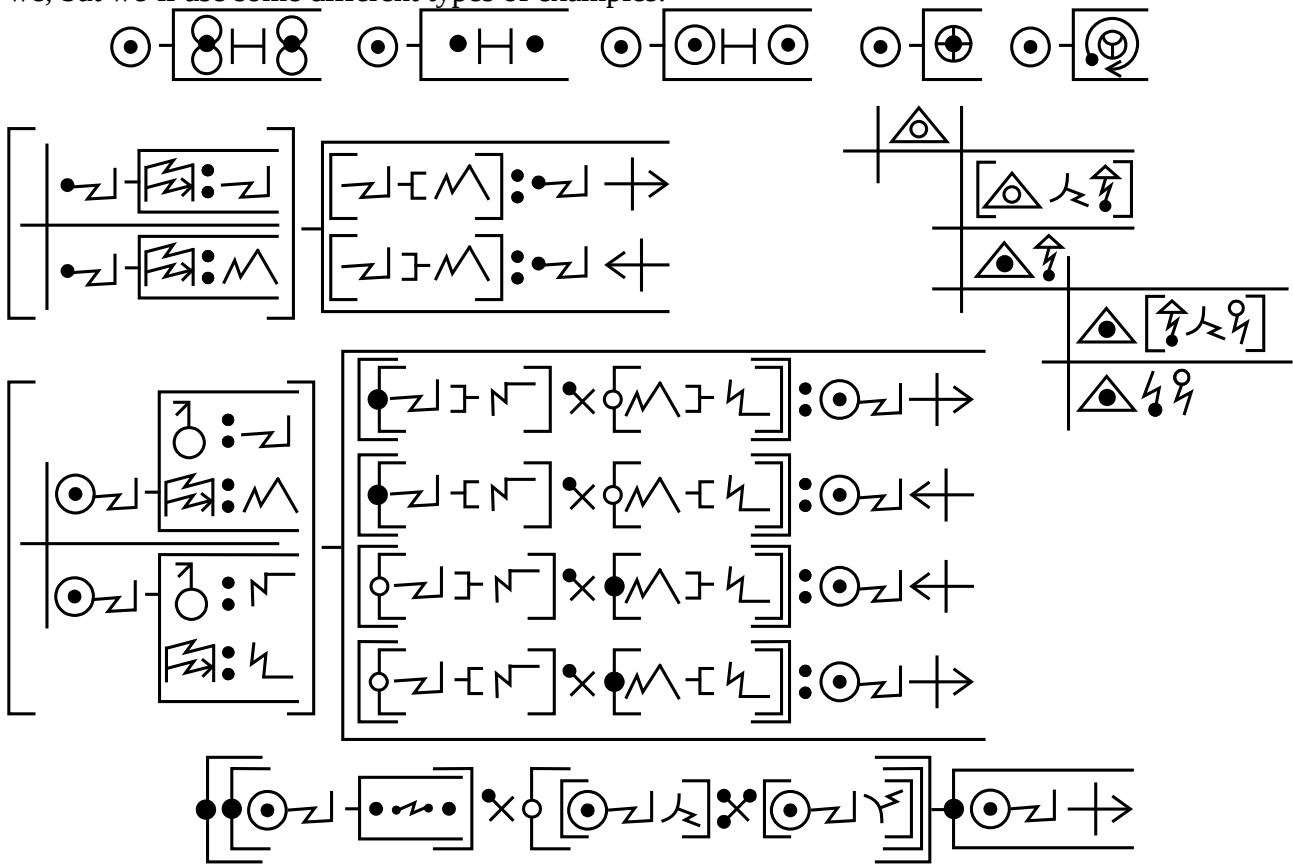
Top line :   system sub(bonded molecules)   system sub(bonded particles)   system sub(bonded systems)
                  system sub(astronomical body)   system sub(star + orbiter / solar system)
        *here we expand our examples of system to clarify how broad the term is
Second line
        left : (past particle energy=a, future particle energy=b)
                sub ((a>b)=particle entropy increase,   (a<b)=particle entropy decrease)
        diagonal : A neutron decay time-line showing entropy steps when decay occurs
Third line: (past system-a sub(volume=a, energy=b), future system-a sub(volume=c, energy=d)) sub{
                if(a<c) and if-not(b<d)) system-a entropy increase *volume increase energy not increase
                if(a>c) and if-not(b>d)) system-a entropy decrease *volume decrease energy not decrease
                if-not(a<c) and if(b<d)) system-a entropy decrease *volume not increase energy increase
                if-not(a>c) and if(b<d)) system-a entropy increase *volume not decrease energy decrease
                }
Bottom Line :   if(if(system sub(particle interact particle) and if-not(system radiate OR system absorb)))
                    then{system entropy increase}

The bottom line is perhaps one of the most important for our future definitions and perhaps the most controversial. Assuming that all interactions within a system lead to its entropy may seem intuitive (to me anyways), but this can start a wild debate, especially when discussing the strong force or gravity. Eg. proton decay is still debatable, protons have internal interaction, but we are not certain they ever decay naturally.

Please understand that this is NOT entropy as you may know it, this is a NEW TERM. It is very similar to entropy, but it also carries the meaning of "the natural internal progression of a system without external interactions"

Possibilities

Our expressions are starting to get complex and even a little abstract now so we need a way to discuss things in a less determinate way, "is equivalent to" and "is a subset/superset of" are no longer enough, we need to be able to say things like "is possible", "is undetermined", "could be one of x possible states/outcomes". For this we will use the wave symbol to represent undetermined/unknown and the particle symbol to represent determined/known.

This may seem like a direct use of quantum mechanical concepts, and it definitely fits the QM language to some degree, but waves are not restricted to quantum mechanics, fluids have waves, there are pressure waves, there are even gravitational waves, and all of them have a similar trait, namely "their form/density/intensity/etc.. changes as they propagate through their medium, time & space", whereas particles, like a proton, "maintain most of their characteristics as they travel through a medium, time & space" (eg. waves usually expand and disperse over larger and larger areas/volumes, particles maintain their density and volume). We will draw off these properties, particles are "fixed/known" and waves are "fluid/unknown".
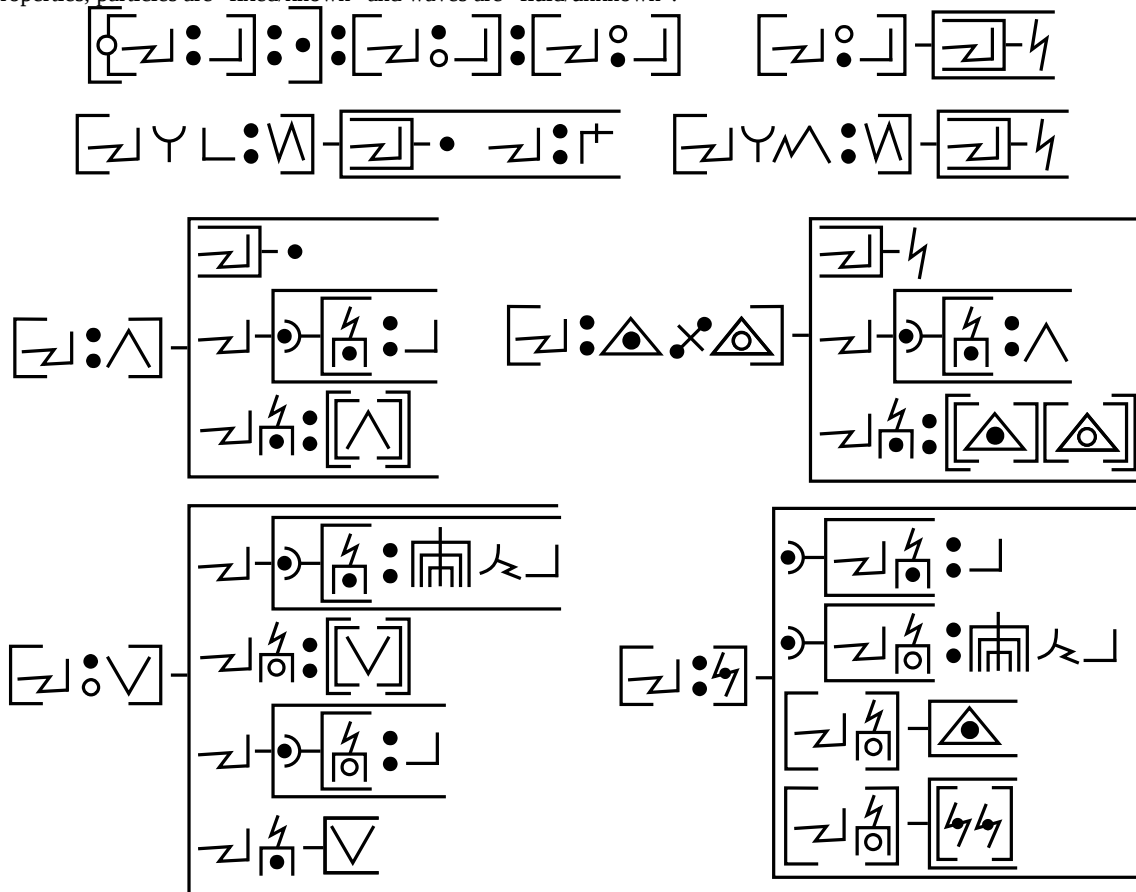
Top line : (inv-eval(a=1)=TRUE) = (a not-equal 1) = (a not-equal 1)

       (a!=1)sub(a is a subset of "wave") *"wave" can represent undefined/unknown

Second line : (a+3=10) sub(a is a subset of "particle"   a=7) *"particle" can represent defined/known

       (a+b=10) sub(a is a subset of "wave")   *a could be anything so it is unknown

    *next I will introduce a symbol for (im)possibilities. It is drawn as a combination of "wave subset particle" implying it a "known subset of a wave/space" in other words "a (im)possible state". The symbol is defined in the examples so i's not necessary for the symbol or for it to be intuitive, I just like to make them as symbolic as I can.

Third line :   (a=2) sub(a is known, a sub(count(possibility)=1), a possibility=array(2))

       (a=proton XOR neutron) sub (a is unknown, a sub(count(possibility)=2),

                     a possibility = array(proton, neutron))

Fourth line :   (a!=5) sub(a sub(count(possibility)=infinity-1),   a impossibility = array(5)

         a sub (count(impossibility)=1),   a possibility sub(5))

       (a=photon) sub(count(a possibility)=1,   count(a impossibility)=infinity-1

        (a impossibility) superset(neutron), (a impossibility) superset(2 photons))

*Yes, I know infinity-1=infinity, here we only write that to express "everything/every possibility except 1".

**Remember, the particle-dot and space-circle are also used to represent charge/lack of charge, it is that meaning we are drawing from to create the symbols for possibility/impossibility.

Times, Periods & States

Before we can finally move on to life we need to be able to discuss some time and state related concepts.
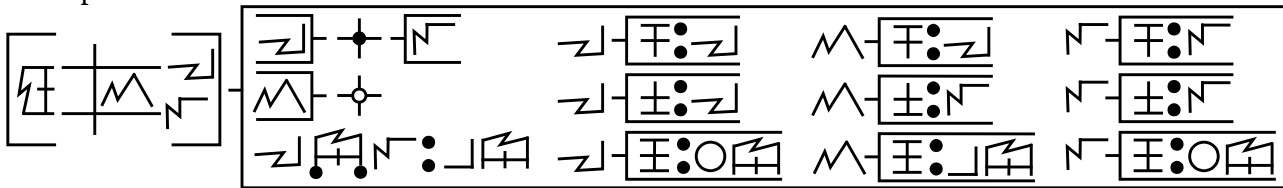
Above we have have a simple time-line on the left.

There are 2 time points labeled and one time period labeled as being 1 Uscript time unit long.
On the right in the subset (left to right in vertical stacks):

Leftmost :   we define a symbol for "time point" and declare that a and c are subsets of it
                 next we define a symbol for "time period" and define b as a subset of it
                 below we state that "a time-interval b = 1 Uscript time unit"
Second :   a sub (period-start = a)   a sub (period-end = a)   a sub (time-line = 0 time units)
Third :   b sub (period-start = a)   b sub (period-end = c)   b sub (time-line = 1 time units)
Rightmost :   c sub (period-start = c)   c sub (period-end = c)   c sub (time-line = 0 time units)

These allow us to discuss time periods and their ranges. It also allows us to refer to specific points in time and distinguish them from periods of time. This also gives us symbols we can use for start/begin and end/finish (period-start/period-end).

　　　*we borrowed the same structure that we used for measuring distance so it shouldn't need definition

Previously we have been drawing states inside the time period window, this has been easy to do so far because we never give concrete times eg. for neutron decay we drew states and processes inside time period windows, but because we never labeled each window with a time frame we so didn't need to be precise, know how long the process took or how long a specific state was maintained.

Above there is a time-line with 3 periods and 4 time points labeled. The 3 periods all have different lengths of 2 time units, 7 time units, and 4 time units.

Below the time-line there are 2 symbol equivalences, these are just to make them easier to draw.
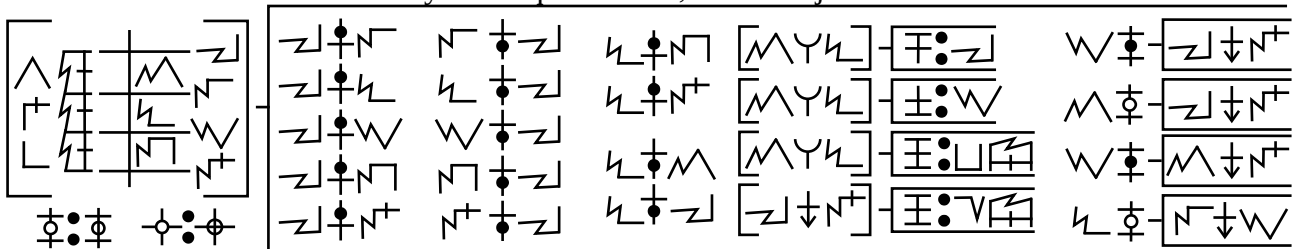
On the right in the subset : (left to right in stacks)
First column: a before b,   a before c,   a before d,   a before e,   a before f,   a before g
Second column :   b after a,   c after a,   d after a,   e after a,   f after a,   g after a
Third column :   d before f,   d before g,   d after b,   d after a
Fourth column :   (b+d)sub(start = a)   (b+d)sub(end=e)   (b+d)sub(time-line=9 time units)
                           (a to/time-arrow g)sub(time-line=13 time units)
Fifth column :   e time sub(a to g), b period sub(a to g), e time(b to g), d period sub(c to e)

*notice there is no "a before b" or "b after a", this is because in graphic before this we defined that b sub(start=a). For the same reason there are no refers to "d-c" and "d-e", because "c starts d" and "e ends d".

This gives us terms for before, after, and during for time points and time periods. There are a couple more things things that can be defined, like "time-period to time-period" and evaluations like "if a is before b", "eval(b is during c)" etc…, but this is enough for our current purposes and those are rather intuitive, they can be added when needed.

Next we move on to stages, these are the entropy line equivalent of time-periods, but instead of time ranges they refer to "state range".

When a system is in a specific configuration, or composed of a specific set of particles/sub-units, or has certain characteristics, this is a "state" in Uscript.

When a system holds an approximate "state" over time with changes or fluctuations that are insignificant to the definition of that state, this is a "state-range".
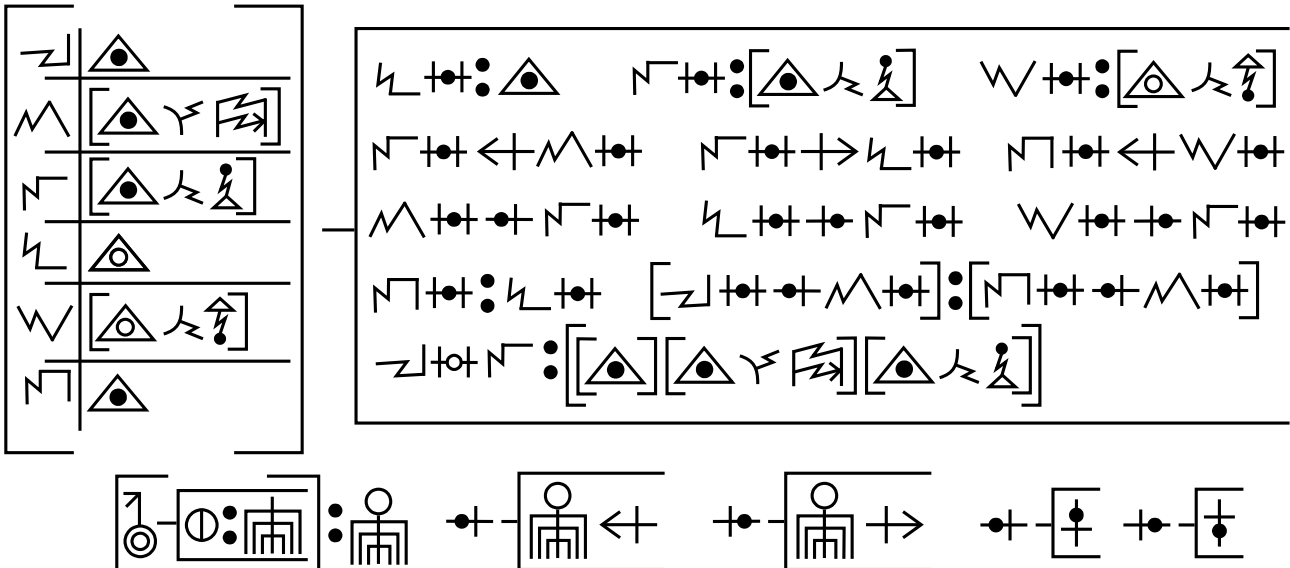
Top left there is a time-line with periods labeled on the left.
Right subset of time-line
       First line :   a state = Proton   c state=(proton emit W+)   e state=(neutron emit W-)
       Second line :   c state decrease-entropy b state
               *in other words "going from c to b decreases entropy" or "state c has less order than state b"
               c state entropy-increase a state
               *in other words "going from c to a increases entropy" or "state c has more order than state a"
               f state entropy-decrease state e
       Third line :   b state precedes c state   d state succeeds c state   e state succeeds c state
       Fourth line: e state = a state   (a state precedes b state)=(e state precedes a state)
               *This line will probably get deprecated. It was meant to divorce states from time but it is perhaps better to create a new symbol for that. The reasoning for it was "there can be near infinite intermediary states and we don't want to have to always reference them all when saying before/after"

       Fifth line : a state-sequence c = array(proton, proton absorbs energy, proton emits W+)
               *this gives us a way to handle state sequences with our array handling tools
      Bottom : (3d spherical space sub(diameter=infinity))=space-infinity *in other words "universe"
          preceding-state sub(Universe decrease entropy)
          preceding-state sub(Universe increase entropy)
          preceding-state sub(before)   proceeding-state sub(after)

The bottom section is not necessary but it is meant to justify/clarify 2 things
     1.The reasoning for preceding and proceeding state using the entropy line is that states changing may not imply system entropy but it does imply entropy of the universe.
     2.We may divorce states from time in some ways, but a preceding states still always happen first, succeeding states always happen after.

Life-lines

No, we have not come to defining "life" yet, this section is just about life-lines, as in "during the life of this proton it interacted with…." or "the life of a star". A Uscript "life-line" is just a series of states over periods that make up the time-line of a system.

A life-line always has a beginning, an end, and at least 1 period/state. The end can be undefined but they still have have one with a value of undefined.

In this example 2 systems combine to form a new system, which exists for a few periods/states, and then divides into 2 systems.

In some cases 1 system may become 2 systems but you may want to still consider the original system to be one of the 2, in that case it is best to use radiate instead of divide. In the case of, for example, human cell devision, I would argue it is division, and the 2 new cells should be considered new systems, I suppose you could argue that one is the original system by using some logic along the lines of "I am following one side of the DNA strand, so one is an original and one is a copy".
*I have doubts that the "same side of the DNA can be followed" like that, but I'll leave that debate to experts.
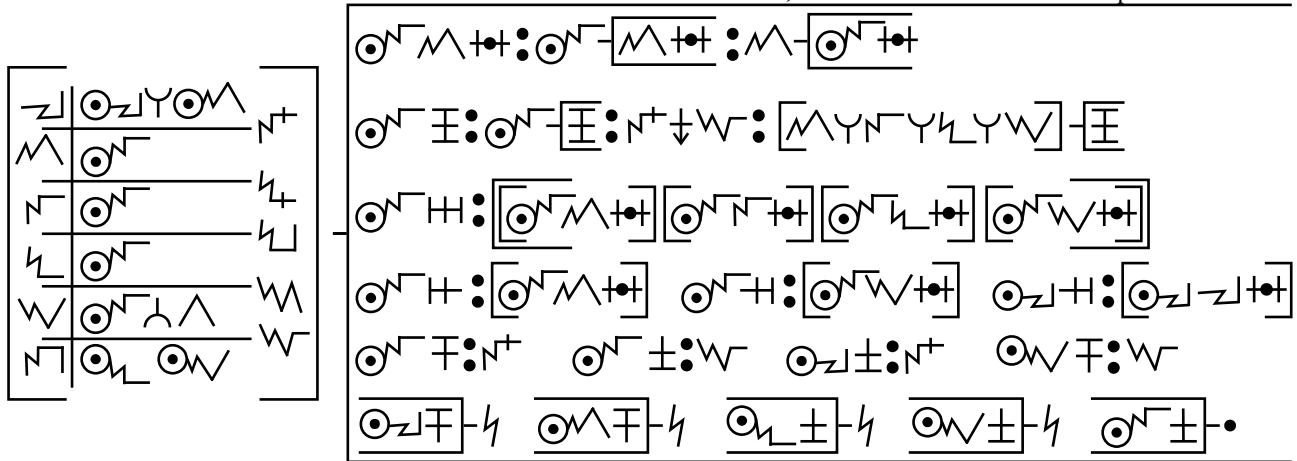
On the left there is a time-line with a periods and time points labeled.
On the right there is a large subset with statements about the time-lines

      First line :   system-c b state = system-c sub(b state) = b sub(system-c state)
      Second line :   system-c time-line=system-c sub(time-line)=g to k=(b+c+d+e) sub(time-line)
      Third line :   system-c life-line = array(system-c b state , system-c c state,
                                  system-c d state, system-c e state)
      Fourth line :   system-c first-state=(system-c b state)   system-c last-state=(system-c e state)
                                system-c last-state=(system-a a state)
      Fifth line :   system-c start = g   system-c end = k
                                system-a end = g   system-e start = k
      Sixth line :   system-a start is unknown   system-b start is unknown
                                system-d end is unknown   system-e end is unknown
                                system-c end is known

How you define a system is completely open, the difference between a system ceasing to be the same system and a system that is just in a different state is rather arbitrary. Am I the same system I was last year? If you replace all the parts on your car is it still the same system? This is a question for philosophy, and when we reach the realm of philosophy Uscript backs off and says "arbitrary!". It is up to the user to define this, which usually depends on the context of the matter at hand, eg. if I am referring to our sun in the context of stars in general, it may cease to be the same system when fusion halts/becomes negligible, but in another more specific context it may cease to be the same system when it expands into a red giant or when its core fusion starts producing a certain element.

Expansion, Contraction, Explosion & Implosion

We will want to discuss life and growth so the concepts of expansion, contractions, explosion and implosion will be very useful. We will define these quite simply as exponential and liner changes in volume. Expansion and contraction are linear, and explosion and implosion are exponential. If you film an explosion (anything from dynamite to a supernova) and slow down the playback speed enough, it looks just like a slowly expanding ball, so using a speed or rate is arbitrary, thats why we categorize based on linear/exponential.

I want to use comparison of positive and negative numbers, sub/super set are the greater-than less-than. 2 is a subset of 3 because 2+1=3, but if you allow negative numbers this logic breaks down, we COULD just define "1 sub(-1) , -2 sub (-3), etc..", but the subset symbol we use is also used for general purposes, in what way is "an anti-electron less-than/a subset of an electron, but an electron greater than/a superset of an anti-electron"?.. so this approach just "makes my brain itch". Lets deal with this number line problem before we begin.

Line 1:   absolute(-1)=1   absolute(2)=2   absolute(7-15)=8
Line 2: eval(a signed-less-than b) = eval((abs(a) + abs(b) + a) less than (abs(a) + abs(b) +b))

First line :   (a^(-b)) = (1/(a^b))   *here we add negative exponents
Second line :   (a possibly sub(b)) = a possibility(b)   (a impossibility sub(b))=a impossibility(b)
                              *just a more efficient way of writing about possibilities
Middle Section : system-a sub(time-period sub(

                                                    period-start system-a volume = a
                                                    period-start < b < period-end
                                                    b sub(volume) = (((start time-distance b)^c)*d)
))
Bottom sub-section of the time-period statements:
   Line 1 : possibility((c=1)AND(d>0))=(period state) super(volume expansion)
          possibility((c=1)AND(d<0))=(period state) super(volume contraction)
   Line 2 : possibility((c>1)AND(d>0))=(period state) super(volume explosion)
          possibility((c>1)AND(d<0))=(period state) super(volume implosion)
   Line 2 : possibility((0<c<1)AND(d<0))=(period state) super(volume reverse-time-explosion)
   possibility((0<c<1)AND(d>0))=(period state) super(volume reverse-time-implosion)

The symbols for expansion and contraction show states changing between a particle and a system, this symbolizes something small and something larger but contained.

The symbols for explosion and implosion show states changing between space and a particle, this can be interpreted as "an explosion leaves only space behind" or "an explosion expands so much that the resulting system has negligible density" or "an explosion radiates everything away leaving only space". The same logic works in reverse eg "an implosion condenses a system so much that if it's new state is seen as a particle, it's previous state was empty space by comparison"

These symbolic representations are because I try to fit symbolic representation and intuitive meanings as much as possible but they could be arbitrary symbols. They are not part of the definition, the definition is in the equations.

The image below uses non-Uscript graphs(our standard graphs on earth, time is horizontal, value is vertical) it is just for readers of this document. If you wanted to include it in a Uscript key you would need to use the Uscript graphs and make them subsections of their respective symbols.
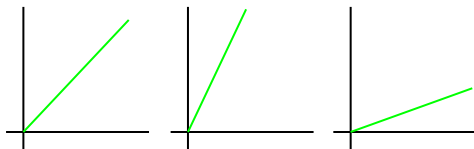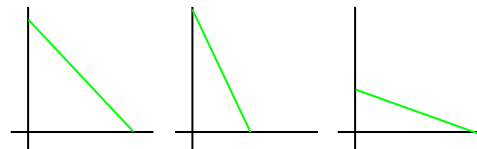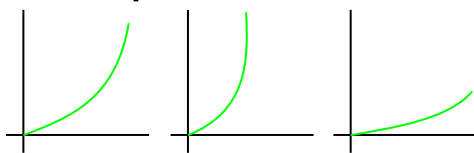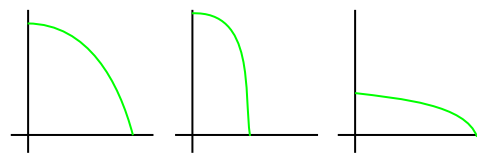

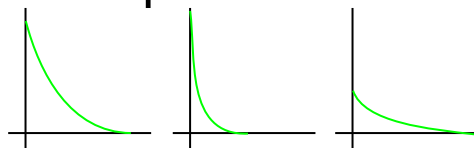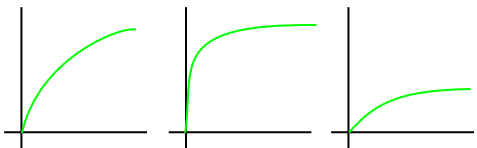"Time reversed" just means "from a backwards time perspective"

If you take an explosion and flip the time axis (horizontal axis in the above graphs) it will look like a "Time reverse explosion".

This gives us access to 6 types of curves.

S curves can be created by combing 2 curves.
Spikes can also be created by combining 2 curves.

If you wanted to describe a specific curve, simple or complex, you would need to do the same as we do now, in other words either draw the curve on a graph or describe it with an equation.

Birth, Death, and Events in between

Now we can put together the concepts we have created to make some pretty complex and more linguistic statements, let's try discussing birth, death and life-events.
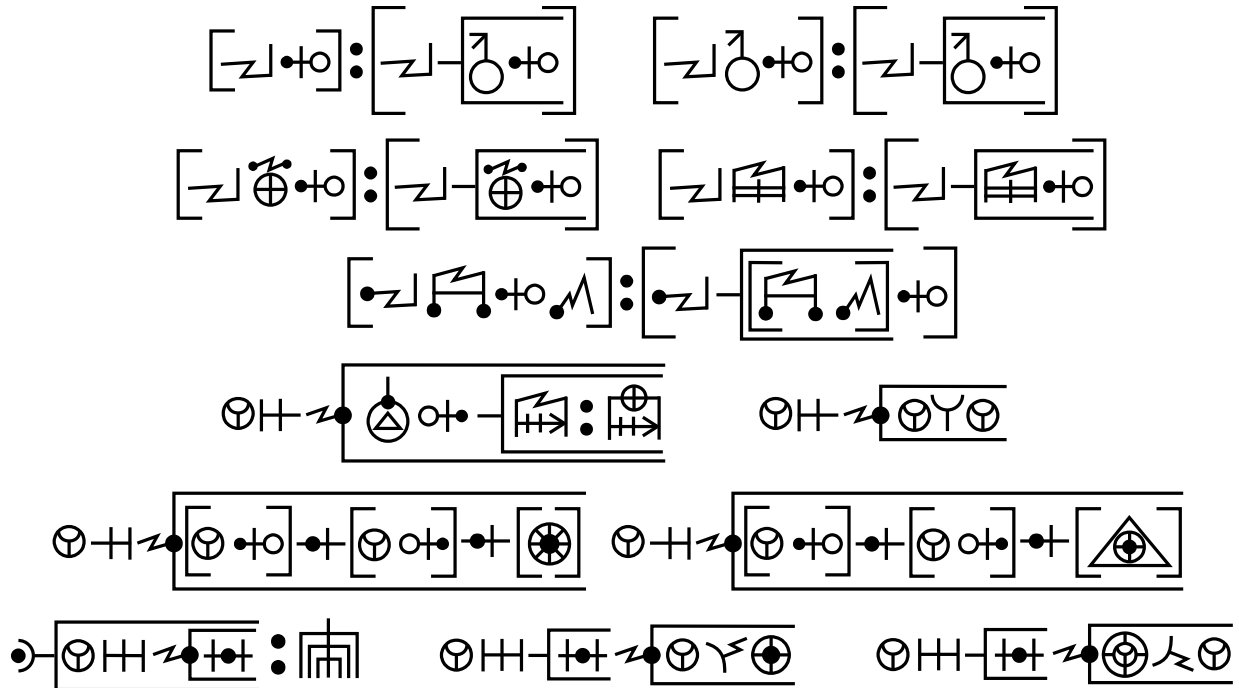
First line :   (a explosion) = (a sub(volume explosion))
                   (a volume explosion) = (a sub(volume explosion))
                          *this first line clarifies that explosion means volume by default

Second line :   (a mass explosion) = (a sub(mass explosion))
                   (a velocity explosion) = (a sub(velocity explosion))
                          *this line shows how to use curves to describe other properties

Third line :   (particle-a distance explosion particle-b)=(particle-a sub((distance particle-b) explosion))
                          *another example, this one is how to use curves to describe changing distances

Fourth line :   star birth possibility (gas implosion sub(force=gravity))
                          *"a star can form from gravitationally collapsing gas"

                   star birth possibility (star merge star)
                          *"a star can form from merging stars"

Fifth line :   star death possibility((star explosion) before (star implosion) before (black hole))
                          *"a star can die in an explosion, then implode into a black hole"

                   star death possibility((star explode) before (star explode) before (neutron star))
                          *"a star can die in an explosion, then implode into a neutron star"

Sixth line :   count(star life possibility(state))=infinity
                          *"there are infinite possible states/events in a stars life"

                   star life sub(state) possibility(star absorb astronomical-body)
                          *"a star can eat/absorb an astronomical body"

                   star life sub(state) possibility(galaxy radiate/emit star)
   *"a star can be ejected from a galaxy"

These examples all deal with general possibilities for a star birth, death, and life events. We have already defined how to describe a specific system and its specific life-life and birth/death/events.

As you can see we can now make some pretty complex general and specific statements, and they are pretty efficient (do not consume a lot of space or writing). Uscript is getting closer and closer to what we consider a language rather than just a notation system.

Uscript Life

I am going to "skip" chemistry and biology. I love the fields and could go into them in depth, but I will leave them for extended versions and information databases. Why? Because we don't want to assume the life is biochemical. We know they exist in the universe, we are assuming they exist on a scale and dimension similar to ours (They can perceive at least our 3 dimensions+time and they are not entities larger than the universe or smaller than atoms, not saying those are impossible, just outside the defined scope of Uscript). They could be biochemical, or be more similar to what we would call machines, or just exist in wildly different environments with wildly different biologies and chemistries.

We need to define life from the ground up to be all inclusive. Luckily we don't need to create a word "life" because our word "life" is defined in a very "earth evolution centric way". Is a virus alive? Is a prion? What about RNA? According to the standard definition, NO! Because, in simple terms, they don't reproduce internally, but this could easily rule out other potential forms of life. So where do we start? I think the best characteristics to use are homeostasis and reproduction.
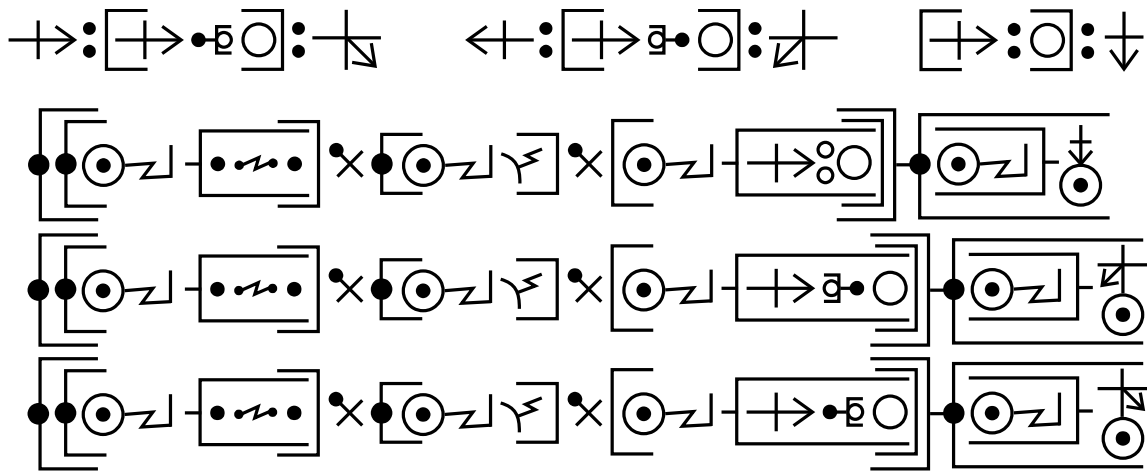
Line 1: entropy=(entropy > 0) =increasing entropy   *entropy can only occur if time passes
         "untropy"=(entropy < 0) = increasing order   *"untropy" new word I created, entropy antonym ;)
         (entropy=0) =subs (arrow of time)   *time flows but no reference to entropy
Line 2:if(system-a sub (particle interaction) AND absorbs AND entropy approx 0)
                then system-a is a homeostasis system
Line 3:if(system-a sub (particle interaction) AND it absorbs AND its entropy < 0)
                then system-a is an "untropy system" *a system of increasing/growing order
Line 4:if(system-a sub (particle interaction) AND it absorbs AND its entropy > 0)
                then system-a is an "entropy system" *a system of decreasing/diminishing order

*notice I use approx equal to zero. We want to include life in this category, and life is not a perfectly stable system, it needs to adapt in some form or another, and may die over longer time scales(like us).

Ok. So we have a Uscript version of homeostasis, and something like formation/growth(untropy) and decay/death(entropy). Untropy vs. homeostasis is a tricky line, because in most examples I can think of we are either decaying, or increasing in overall order but also increasing in mass and volume. There are definitely moments/periods where our potential energy per gram or per cubic cm increases so I suppose these are our "moments of untropy".

So are there true "living untropy systems"? Plants definitely experience untropy when they convert massless photons into higher energy state molecules. I think at the very least we could look at living systems and find "moments/periods of untropy". Regardless, we have the term available and it can be useful for non-living systems too (a growing black hole is an untropy system) and it can also be useful when we soon get to metaphors.

Next we have reproduction. We can already define a growing system, It either gets bigger, or it's sub-system count increases or both. eg. growth for us is an increase in cell size or cell count.

I suppose one can imagine a "non-gowing" being/entity whose behavior we would classify as "alive", but in order for it to never have experienced a growth cycle that would imply it always existed which would be a deity/god. I suppose you could argue it just randomly formed, all at once, or "it was not alive until its final particle came into place", but that is just an argument on arbitrary definitions of system classification. The same argument can be made about any life, eg. At what point does an embryo become an animal?

Reproduction on earth takes many forms, it's actually very hard to classify them and define categories. Reproduction is so vital to life that it has adapted in many ways and comes in many forms that most people can't even imagine. I am not going to wrestle with this area for now. I like biochemistry but biology is not one of my strong suits, and secondly it is hard enough to define reproduction categories that are earth-centric and it could easily end up being dozens of pages long(in short, the details of reproduction are too much for this introduction document).

Here is a simple approach to broadly define reproduction without detailed subcategories for now.
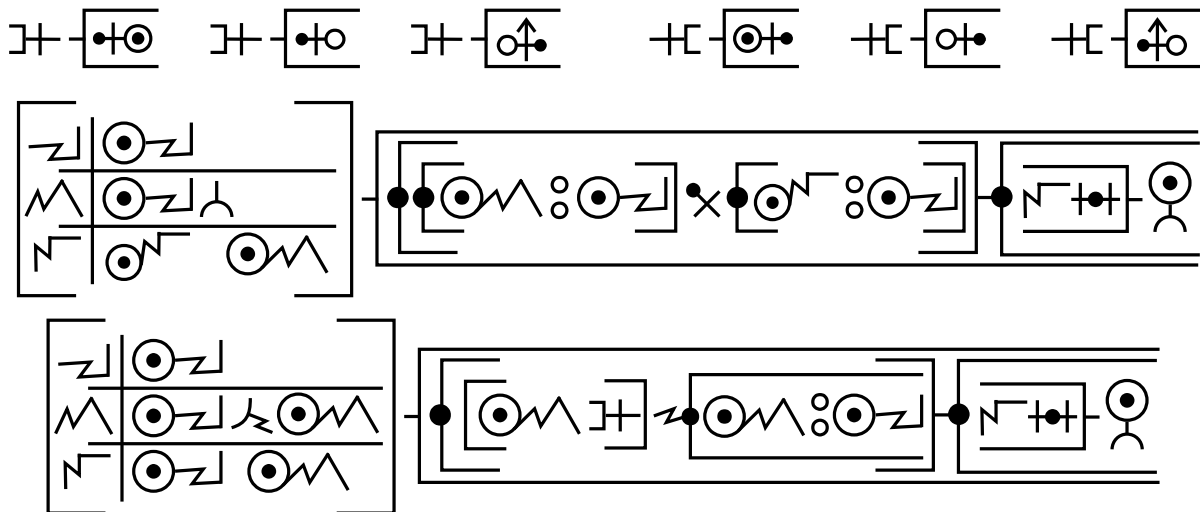
First line : growth sub(expansion)   growth sub(explosion)   growth sub(reverse time implosion)
                  shrink sub(contraction)   shrink sub(implosion)   shrink sub(reverse time explosion)
                              *here we create general symbols for growth/increase and shrink/decrease
Middle section : A time-line that shows a system dividing into 2 new systems with a statement
          Statement : if((sys-b ≈ sys-a) AND (sys-c ≈ sys-a)){ b-state is a subset of reproduction }
                          *if a system divides into 2 systems that are approximately the same as the first it is reproduction
Bottom section : A time-line of a system emitting/ejecting a system
          Statement : if((sys-b growth) possibility(sys-b ≈ sys-a)){b-state is a subset of reproduction}
   *this covers things like eggs, birth of babies, budding, regeneration, fragmentation, etc..

This is a VERY general definition, it can be refined but I intentionally made it a very broad definition. We could try creating broad sub-categories, but it's quite difficult to draw arbitrary categories here and seem too earth-centric. I will leave this for future development.

I used the growth symbol alone, we defined the sub-symbols to default as volume, but have not defined a default for growth. If you don't like this you can define something specific, but it doesn't matter because we use possibility. By using possibility it is true for most things, eg "sys-b time possibility" or "sys-b absorb energy possibility" or "sys-b future state possibility". We don't require that it DOES become a similar/copy system of system-a, just that it CAN. I used growth because it is smaller than system-a, thats part of the "radiate/emit" symbol (There are exceptions eg. math, but otherwise the "wave" side of the symbol is meant to be symbolic of smaller/energy vs the other side larger/matter)

Has/Contains, Minimum & Maximum

Here we add a few more array handling functions and create terms for "contains/has", minimum value, and maximum value. We will also use this opportunity to flesh out our procedural programming a little more. These will be needed in the next sections.
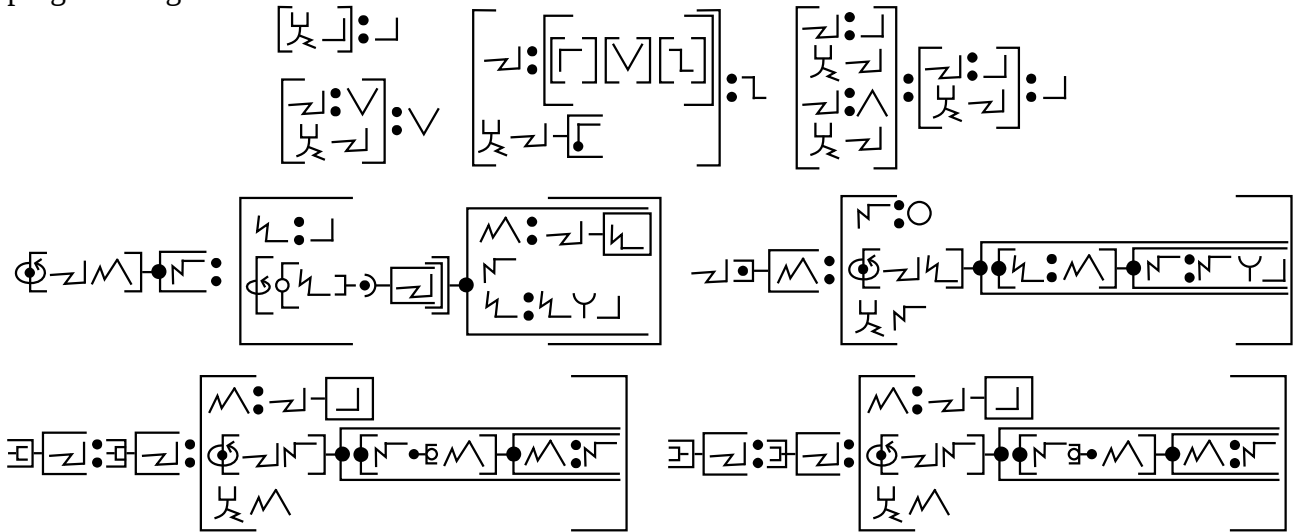
Top row section:

Left : (return 1) = 1   (a=5; return a;)=5

Middle : (a={3,5,12};return a[1])=12

Right : (a=1; return a; a=5; return a;)=(a=1;return a;)=1

Second row section :

foreach(a as b){c} = (d=1; while(d<count(a)) { b=a[d];   c;   d=d+1;})

a contains(b) = (c=0;   foreach(a as d){if(d==b){c=c+1;}} return c;}

Third row section :

max(a)=max(a)=(b=a[1];   foreach(a as c){ if(c>b){b=c;}}   return b;)

min(a)=min(a)=(b=a[1];   foreach(a as c) { if(c<b){b=c;}}   return b;)

This now give us several new useful terms

  Return : This can be abstracted to situations where a complex system returns information/values/reports to its parent system

  Foreach : Can be abstracted to describe process that check through/analyze/sort things

  Contains : Until now we have been accomplishing this using the subset symbol. I feel the subset symbols is being overused, its used for declaring things as subsets, describing qualities, evaluating, comparing, etc.. I think its about time there was a separate way to explicitly say "a contains x b's". It returns 0 if there are none, which will evaluate to FALSE, and the count of matches which will evaluate to TRUE if there are any. We could also try defining this with something along the lines of "count (subset(element=x))".

  Min/Max : Very useful concepts, also useful for talking about ranges, limits, etc.. .

*Remember, Uscript starts array count at 1 not 0. I know this irks most programmers, me too, but the 0 symbol doesn't work well as a combination symbol because its reserved for so many other uses.

**the return symbol is a combination of "superset" and "radiate". radiate into the superset.

The further we get the more choice we have in how we define a term. Terms could be defined in various ways and each way can have tons of different possible expressions. Our vocabulary is growing larger and if we abstract terms it increases our available choices even more.

Subspace, Subsystems, Environments and Neighbors

To discuss life we will need to be able to discuss environments. All life depends on an environment to supply it with what it needs to survive. We also need to be able to discuss the internal space and workings of an organism.

For this we will use 4 main concepts.

   Subspace – A space within a system or space

        Subsystem – A system within a subspace

        Environments – The space surrounding/containing a system or space

        Neighbors – A system within an environment

Subspace and subsystem are easy, but environment and neighbors are a bit arbitrary, technically the entire universe is my environment and everything within it is my neighbor, but this is a rather useless definition, so where do we draw the line? Interaction is the key, potential interaction defines environment while potential bilateral interaction defines neighbors. This still means there are arbitrary factors based on interaction speed and system lifetime. My maximum environment is a sphere radius "speed of light * my lifetime" and my maximum neighbor distance is "(speed of light * my lifetime) / 2. To apply these concepts usefully we still need to add arbitrary thresholds of course, and that must be done on a case by case basis.
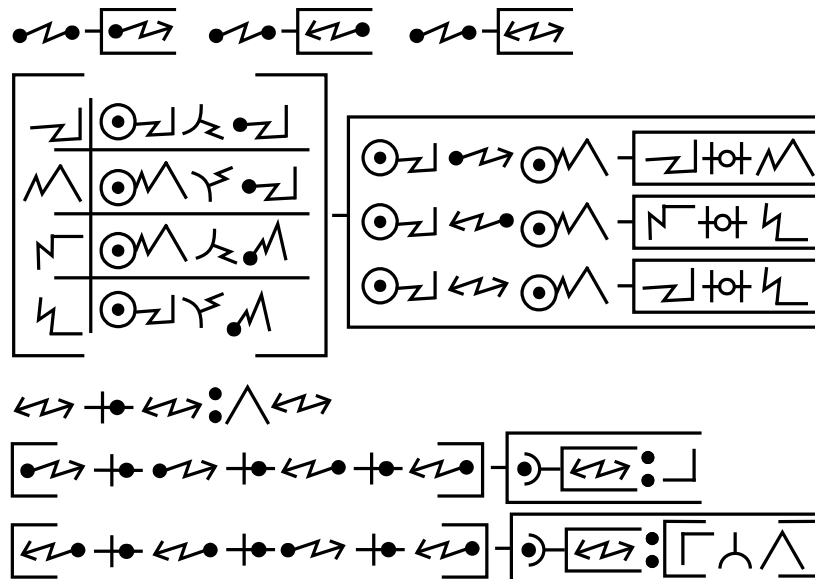
First line :   interaction sub(send-interaction)   interaction sub(receive-interaction)
                    interaction sub(bilateral-interaction)

Time-line of bilateral interaction:

        Statements in subsection:

                sys-a send-interaction sys-b sub(a state-until b)

                sys-a receive-interaction sys-b sub(c state-until d)

                sys-a bilateral-interaction sys-b sub(a state-until d)

   *si=send-interaction   ri=receive-interaction

Third line :   bilateral interaction after bilateral interaction = 2 bilateral interactions

Fourth line :   si after si after ri after ri sub(count(bilateral-interaction) = 2)

Fifth line :   si after si after ri after si sub(count(bilateral-interaction) = 3/2)

Above we define sub types of interaction, "X send-interaction Y " means X emits and Y absorbs, "X receive-interaction" means Y emits and X absorbs, and a bilateral interaction is bilateral sequence of interactions. For bilateral interaction the count only increases by 0.5 every time an interaction switches directions.

If you are particularly analytical and observant you may be wondering "that definition doesn't cover the case where interactions of different direction overlap in time". For example, I could be receiving a constant stream of photons from the sun, and reflecting a constant stream of photons back, so let's suppose I am being hit by 1 photon every second and emitting one photon every second, does that mean the counter increases every second? The answer is NO. Bilateral interactions are counted based on completed interactions only, and this detail is cleared up in these next statements.
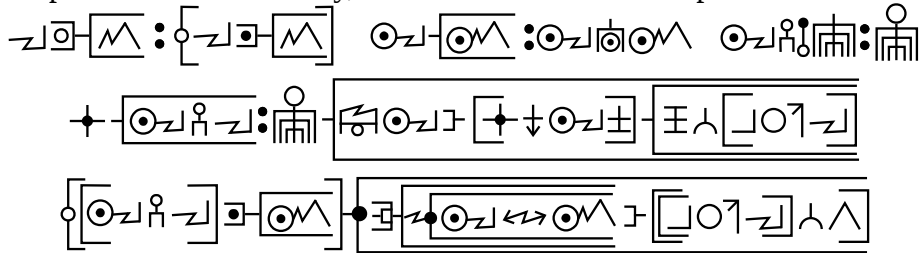
First line :   a not-contain(b) = not(a contain(b))   sys-a sub(sys-b) = sys-a subsystem sys-b
                 sys-a environment -infinity = universe
Second line : time sub(sys-a environment a) =
                 universe sub(distance sys-a < (time until sys-a end)sub(time-line/(0x10^a)))
                         *notice we used the space-between measure instead of distance between center of mass
Third line :   if not(sys-a env a) contains(sys-b)
                         then(max (possibility(sys-a bilateral interaction sys-b)) < ((0x10^a)/2))
Fourth line :   sys-a subspace = sys-a sub(space)   not(sys-a subspace volume > sys-a volume)
                 sys-a environment = sys-a environment 0
                 (sys-a neighborhood a)sub(volume) = (sys-a environment (a/2))sub(volume)
Fifth line :   (sys-a env=a   sys-a env contains(sys-b)   sys-a neighborhood not-contain(sys-b))
                 =((sys-a env a) contains(sys-b)   (sys-a neighborhood a) not-contains(sys-b))

The first line gives us a symbols for doesn't-contain, subsystem and environment.

The second line gives us a way of restricting the size of the environment. An environment size of "0" means "a sphere the size of the speed of light * the remaining lifetime of the system, which means "environment 0 is everything in the universe capable of interacting with that system during it's lifetime, environment 1 is a sphere 1/0x10 the size, environment 2 is 1/0x100 the size, etc..

The third line says "the maximum number of bilateral interactions with something not within an environment is the (range of the speed of light/the range of the environment)/2" This clarifies that bilateral interactions require twice the distance and time to occur than single interactions and that each directional interaction must complete before the switch in direction interaction is counted.

The fourth line defines subspace, a term that will need some clarification later as its meaning and usage here is not extremely clear, it also sets the default value for environment as 0 and clarifies that the neighborhood has half the range as the environment.

The fifth lines says we can set a system sub variable named environment to establish a new default environment size.

The environment vs. neighborhood is a bit arbitrary, they are just 2 types of of environment, but I think it is a significant distinction. A range for things you could influence or be influenced by, and a range for things you could communicate with.

All, None, and Some

Just adding a few more terms. We already have ways of expressing all, none and some using procedural logic, but these are rather important and frequently used terms so lets make some more efficient notation for them.
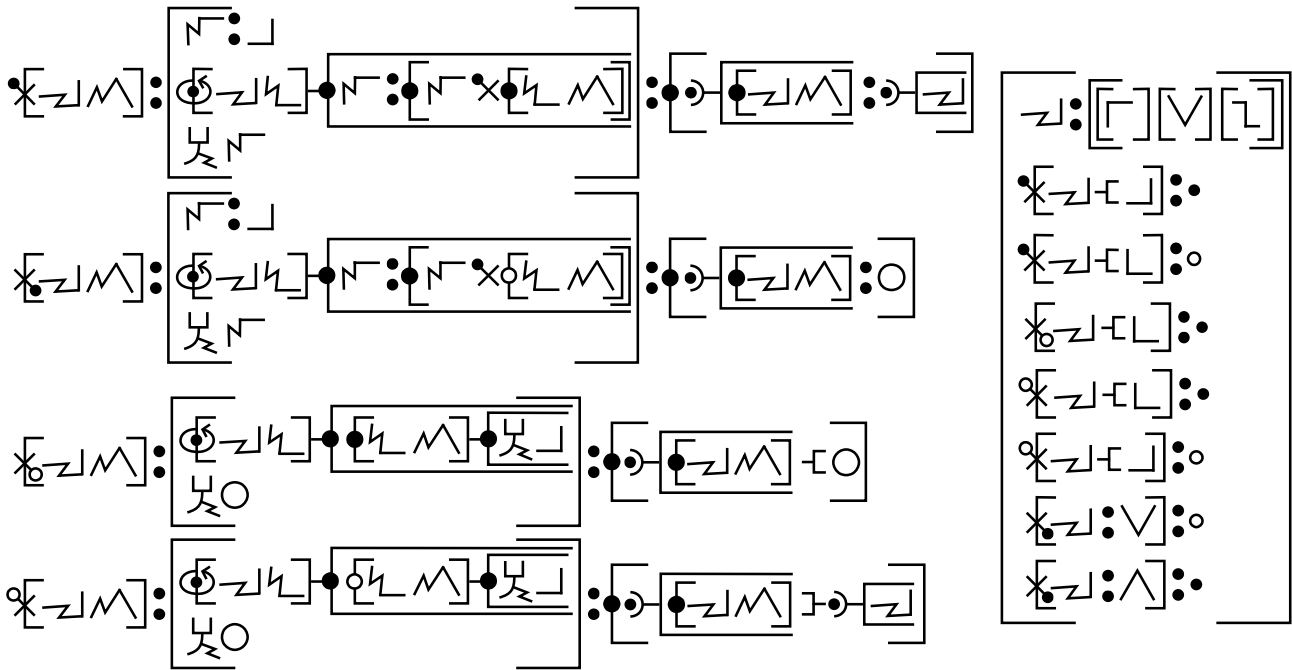
Row 1: all(ab)= (c=1; foreach(a as d){c=c&&(db);}return c;) = eval(count(eval(ab)) = count(a))
Row 2: none(ab)= (c=1;foreach(a as d){c=c&&!(db);}return c;) = eval(count(eval(ab)) = 0)
Row 3: some(ab)= (foreach(a as d){if(db){return 1;}}return 0;) = eval(count(eval(ab)) > 0)
Row 4: some-not(ab)=(foreach(a as d){if(!db){return 1;}}return 0;)=eval(count(eval(ab))<count(a))
Right side : (a=array(3,5,12)

        all(a>1)=TRUE   all(a>4)=FALSE   some(a>4)=TRUE   some-not(a>4)=TRUE

        some-not(a>1)=FLASE   none(a=5)=FLASE   none(a=2)=TRUE   )

These definitions give use clearly defined simple and efficient ways to use all/none/some.

The Expressions we used not only define the terms, but also clarifiy some procedural and array handling methods. Most importantly the third expression in each row shows a simple way of running evaluations on each element and counting the results.

Technically out "some" means "at least 1". If you want to say "at least x" then just define that with expressions using count or contains.

These definitions don't just allow us to save space, they also give use very useful terms that we can abstract later.

Conversion & Consumption

Another key trait of living systems is that they eat. Living systems need to fight off entropy so they must consume energy. All examples of life I know of absorb energy in some form and convert it into other forms for usage. Plants absorb light and convert it into carbohydrates, all life converts carbohydrates and other chemical compounds into more useful forms like ATP. It is not impossible to imagine a life form that does not need to convert the energy it absorbs from it's environment, but this can just be called pure absorption which we already have a term form.
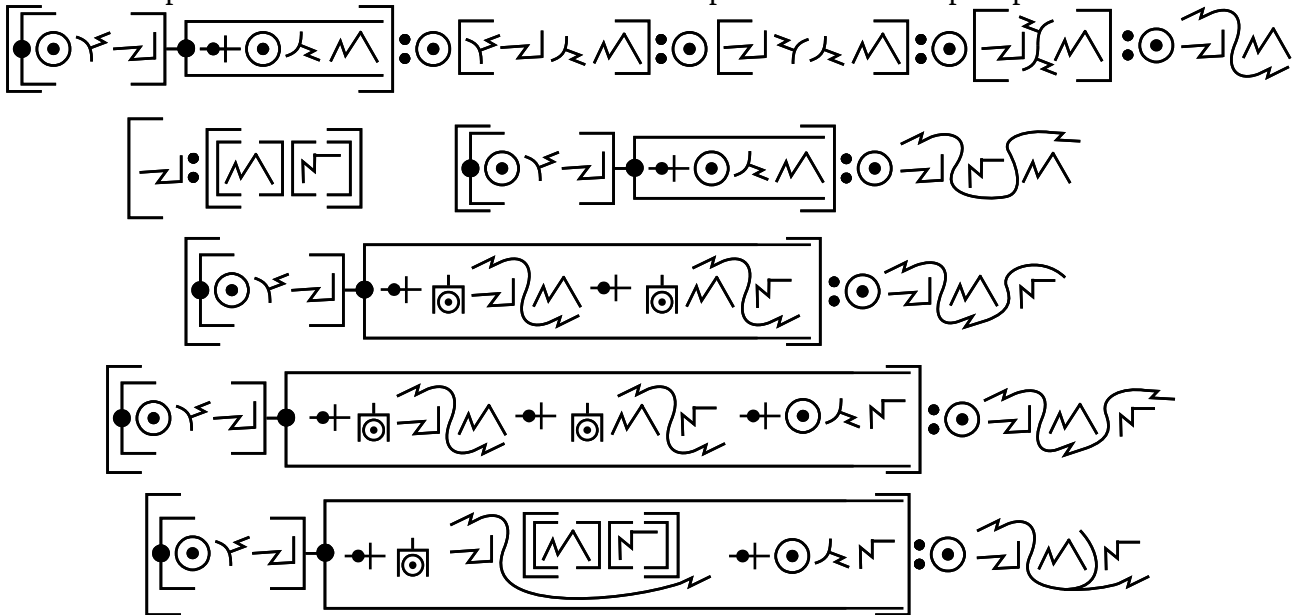
In Uscript we will have 3 main concepts

        Absorption – simply absorbing something like energy or matter.

        Conversion – absorbing one thing and excreting another

        Consumption – absorbing + converting + retaining a portion in some form within the system

These concepts also need to be able to be combined to produce more complex processes.

Top line : (if(sys absorb a) then{next sys emit b}) = sys(absorb a emit b) = sys(absorb a emit b) = = sys(a convert b) = sys a convert b

Second line : (a=array(b,c))   (if(sys absorb a) then{next sys emit b})=sys a retain c emit b

Third line :   (if(sys absorb a) then{next subsys a convert b next subsys b convert c})
              = sys a convert b convert-retain c

Fourth line : (if(sys absorb a) then{next subsys a convert b next subsys b convert c next sys emit c})
              = sys a convert b convert c

Fifth line : (if(sys absorb a)then{next subsys a convert b&c next sys emit c})
   =sys a convert (retain c emit b)

These will allow us to express some of the more complex aspects of systems without resorting to time-lines filled with various labeled particles. We definitely could just use time-lines, pretty much everything can be explained with graphs/illustrations/time-lines, but with Uscript we are trying to build tools that operate in a more linguistic fashion, and this is a good practice/test.

My English descriptions of these complex conversion/consumption symbols is a bit poor, it's hard to match the more graphical nature of the symbols with simple English translations in these examples.

You can also create nested conversion/consumption by placing entire conversion/consumption symbols within in each other. But I wont go into that here, these tools will suffice for our immediate needs. Also note, next just means "a later state" it doesn't have to be immediately afterwards.

Models

Models are descriptions that can be used to understand and predict something, they are usually mathematical description of systems. We are already using models throughout this document. For many higher level concepts, such as intelligence, we need to apply the concept of model more abstractly so we will define a term for it.
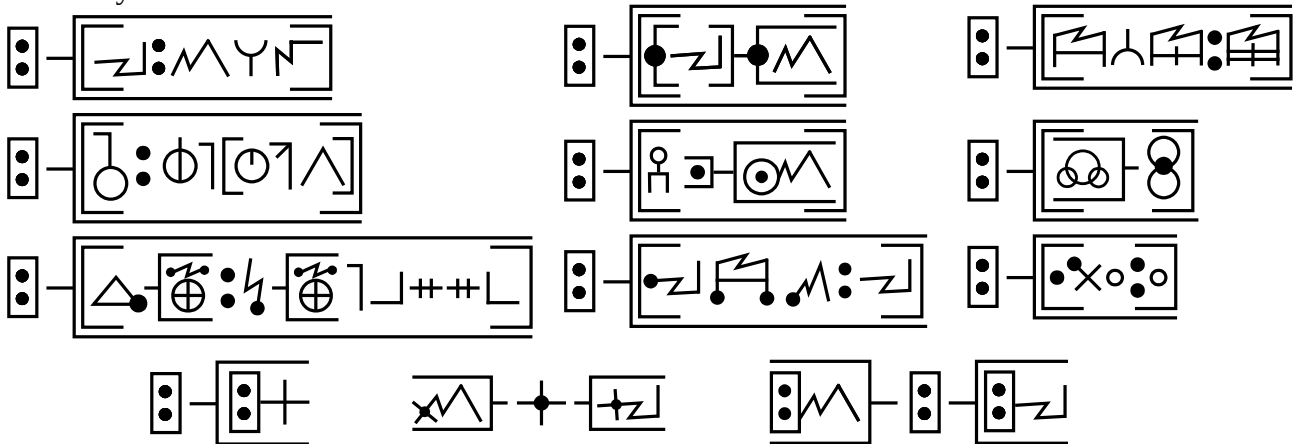
The model symbol is just an equals symbol inside a box(meant to represent brackets) which are 2 of the most important elements frequently used in models.

I won't go through each example given, but here the top 3 rows show 9 examples of models, everything from a simple category definitions and math equations to logical operation to physics models. I think this is enough to get across the meaning that "a model is any form of definition, logic, categorization, formula, etc.."

The bottom line shows a subcategory of model, the model symbols combined with the time symbol, we will get into this category later pehaps, suffice to say for now it means "models that incorporate a time element" which will be defined as models that can predict the future of infer the past. We also introduce a simple symbol for "time point variable" and "model variable" using the same structure we have used many times already of combining the symbol type with a variable.

Evaluating the quality of a model is a tricky business which we will wont get into here.

For evaluating model quality first we would need to define scope and range of the model, we need to discuss accuracy, and we also need to deal with which variables are input and which are outputs. All of these elements are basically dimensions. We need to define tolerance all possible combinations of all possibilities are valid applications of a model, etc..

For now, this can be dealt with by incorporating these elements directly into the model.
eg. Newtons equation for gravity is valid on objects of certain size and mass, moving at certain speeds, at certain distances, in certain external gravitational fields. Even Einsteins field equations for gravity don't apply at the quantum level.

To handle this a model must "return wave/unknown" if it wishes to not be evaluated outside of it's scope. This is accomplished by adding "if(x not in scope){return wave;}" statements to the top of a model. Without these it will be assumed the model is meant to accurate across all scopes and ranges for all possible variables as output.

A model can also involve statement of tolerance rather than an equality
        eg "x > (b^c)-1 && x < (b^c)+1" giving us a tolerance of +-1

This does not mean it is necessary to add all these qualifiers to every model in every case. But if you want to rigorously evaluate and document the quality of a model, this would be needed (regardless whether written in Uscript or any other language).

You can also require variables by adding "if(x==wave){return wave;}" to require that a variable be provided as input.

These aspects will be skipped in this document, but the logic laid out above is pretty straightforward and you already have all the tools should you wish to do it. For now I will be discussing models in more general and abstract terms and will avoid "evaluating model quality".

Next we will apply models and define how they respond under various inputs.
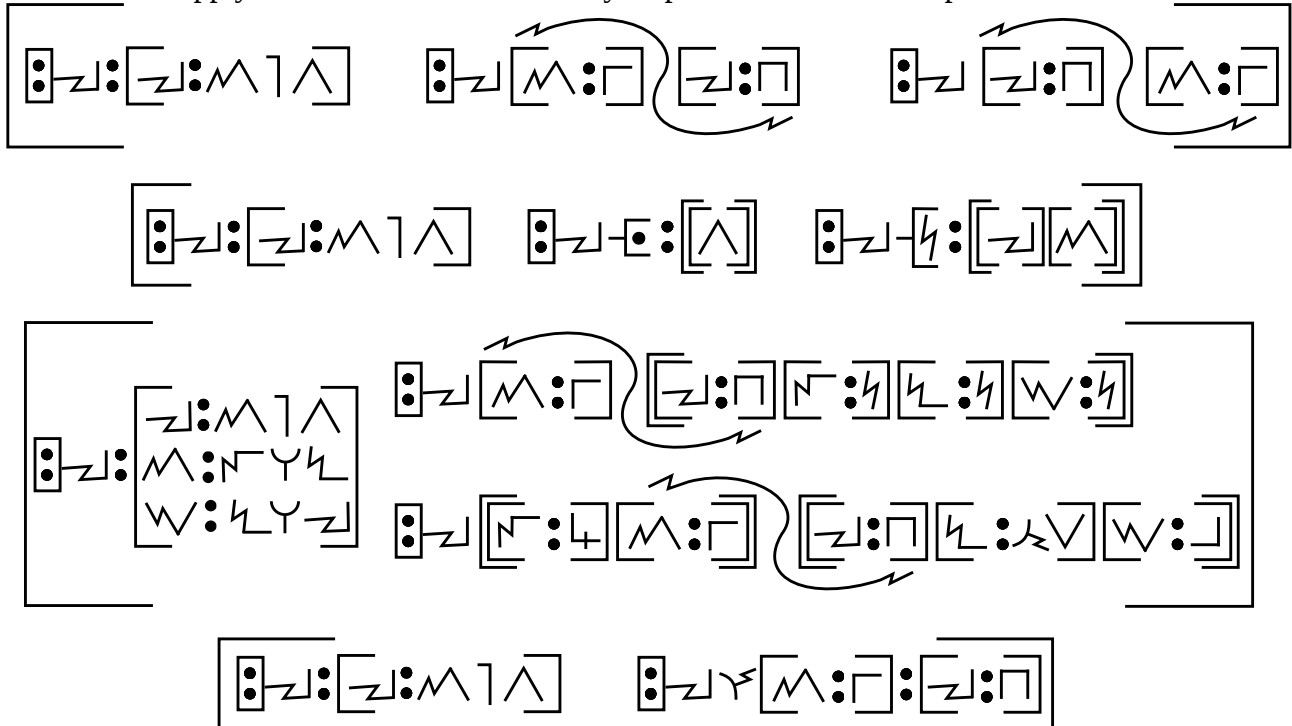
Top section : (model-a = (a=b*2)   model-a (b=3) convert (a=6)   model-a (a=6) convert (b=2))
Second section : (model-a = (a=b*2)   model-a sub(constant) ={2} model-a sub(var) = {a,b})
Third section : (model-a = (a=b*2;   b=c+d;   e=d+a)
                        model-a (b=3) convert {(a=6), (c=unknown), (d=unknown),(e=unknown)}
                        model-a {(c=8),(b=3)} convert {(a=6), (c=-5), (e=1)}   )
Bottom section : (model-a = (a=b*2)   model-a input (b=3) = (a=6))

The top section shows us how to display a model operating(converting inputs into outputs). The convert symbol was designed for physical systems, here we abstract it to models and variables. This notation is useful when you want to describe a model's behavior without defining the model. We do define it here, but we could choose not to define the model and only describe its input and output behavior, this is good when we want to discuss things abstractly or describe a model have are searching for or have not yet defined.

The second section just shows how to access arrays of the constants and variables in a model. The variables array is particularly useful if you want to discuss possible inputs/outputs or different ways to execute the model.

The third section shows how models take and return arrays when inputs/outputs are more than 1. If you are a programmer who hates duck-typing, then just define it so even single element returns are an array like I did with the second section.

The bottom section shows how to execute a model so the result can be incorporated into larger equations and expressions.

Cause, Effect, and Change

Cause and effect are long overdue, so let's finally create those terms.
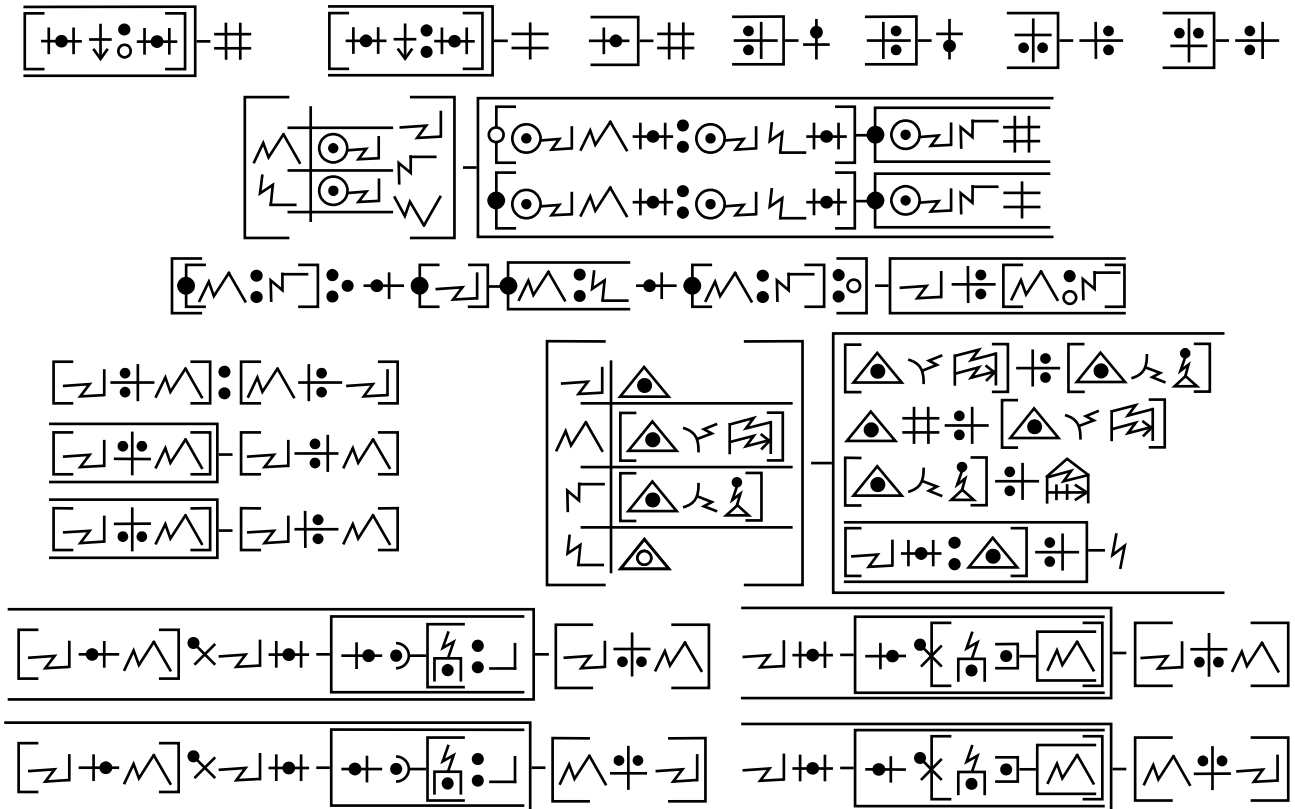
Top line :   (state time+ != state)sub-of change   (state time+ = state)sub-of no-change

                        (future-state)sub-of change   (cause)sub-of past   (effect)sub-of future

                        (inevitable-result)sub-of effect   (only-possible-cause)sub-of cause

Second section: general time-line, with a subsection of statements :

                if(sys-a b state = sys-a d state) then{sys-a c change}

                if(sys-a b state != sys-a d state) then{sys-a c no-change}

Third section :(eval(b=c)=TRUE next if(a){b=d;} next eval(b=c) FALSE) sub(a caused (b!=c))

Fourth section :

      Left : (a because b) = (b cause a)

              (a must-be-because b) sub-of (a because b)

              (a must-cause b) sub-of (a cause b)

      Right : A time-line of neutron decay with sub-statements :

              (proton absorb energy) caused (proton emit W+boson)

              proton change because (proton absorb energy)

              (proton emit W+boson) caused by weak force

              (a state = proton) is-an unknown

Second last line :

      ((a next b) AND a state sub(next-state count(possibility)=1)) sub-of (a must-cause b)

      (a state sub(next-state all(possibility contain(b)))) sub-of (a must-cause b)

Last line :

      ((a before b) AND a state sub(previous-state count(possibility)=1)) sub-of(a must-be-because b)

      (a state sub(previous-state all(possibility contain(b)))) sub-of (a must-be-because b)

This gives us plenty of very useful terms, we can use these to discuss change or lack of change/stability, cause and effect, and easily declare that a certain result must imply a certain cause, or a certain event must cause a certain result.

Uscript Intelligence

We will define intelligence as a combination of the qualities of pro-activeness, awareness, consciousness, and study.

Action, Reaction, Proactive & Reactive

In physics there is no real concept "a proactive action" vs. "a reactive reaction" because from a physics perspective the universe is deterministic (I suppose we can get into arguments about this with quantum mechanics… me I'm fond of a pilot wave theory interpretation of QM which preserves determinism to help me avoid these debates).

Physics only really discusses the concept of action as part of an "action reaction pair", if you look at a single event there was a cause and a effect, an action and a reaction, but that action is itself a reaction to a previous action, and so on until the big bang(that's just as far as we can infer). The same rules apply for conscious life. My actions are all reactions, pure pro-activeness is non existent but we can define a grading system.

We will define all events as "actions", a "pure reaction" is one where "the environment interacts with a system and there are no internal interactions before the system reacts with the environment". Each reaction can also be called an "action", actions have an "action value" which is 1 + the number of internal interactions that chained between external input action and the system's reaction into the environment"
*For now I don't discuss the complexity of reaction chain branching, which branch to use or their relative significance
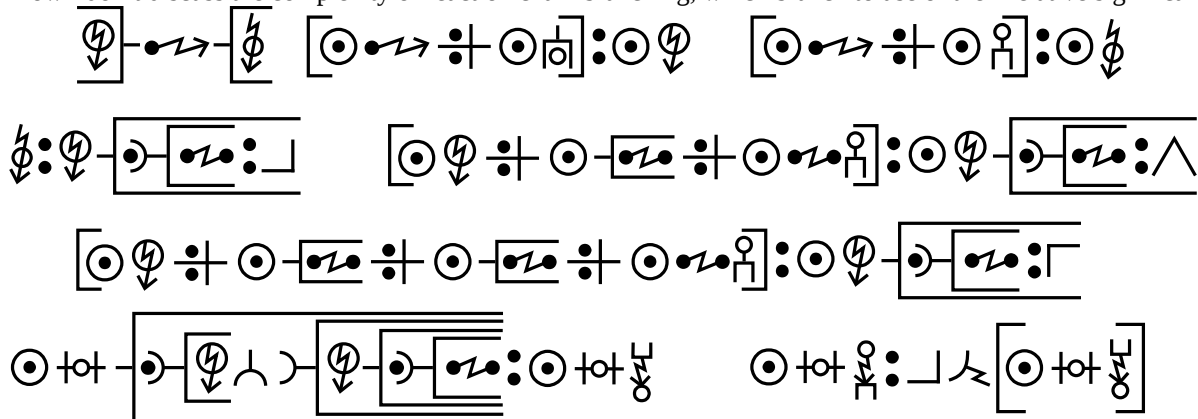
Line 1 :   (action) sub-of directional-interaction sub(reaction)
  (sys outgoing-interaction caused by sys subspace) = sys action
  (sys outgoing-interaction caused by environment) = sys reaction
Line 2 :   reaction = action sub(count(interaction)=1)
  (sys reaction caused-by sys sub(interaction) caused-by sys interact environment)
    = sys action sub(count(interaction)=2)
Line 3 :   (sys reaction caused-by sys sub(interaction) caused-by sys sub(interaction) caused-by
    sys interact environment) = sys action sub(count(interaction)=2)
Line 4 :   sys time-period sub(count(action) / sum(action sub(count(interaction))))
    =sys time-period reactiveness
  sys time-period pro-activeness = 1-(sys time-period reactiveness)

So our definitions are :
  "actions and reaction are directional interactions"
  "internal forces cause action, external forces reactions"
  "all reactions are actions with 'interaction count 1' "
  "'interaction count' is the length of the chain of internal internal interactions +1"
  "reactiveness is number of actions / sum of their interaction count" (is between 0 and 1)
 "pro-activeness is 1 - reactiveness"

Mind, Observation & Awareness

We will define the concept of "mind" as "the subsystem that contains models of the environment". We have not required that it only contain models of the environment, so a mind can still be aware of internal systems, but if it is not at all aware of it's environment or anything outside the system, we don't call it a mind.

Awareness of X is the simple state of having a model that contains X. The validity of the model is not addressed here, that is a longer discussion for another day. Degrees or accuracy of awareness also requires a large amount of definition and is not necessary for now so it will be left out of this first version of Uscript.

Observing X just means receiving an interaction from X caused awareness of X. If you already had a model of X it is still an observation because at the very least you become aware that X is still true, or X still exists, or X has not changed/moved, etc…

Below we define model broadly and include time-point sub-values to clarify that even the state of something at a time can be a model, this basically means that "information is a type of model".
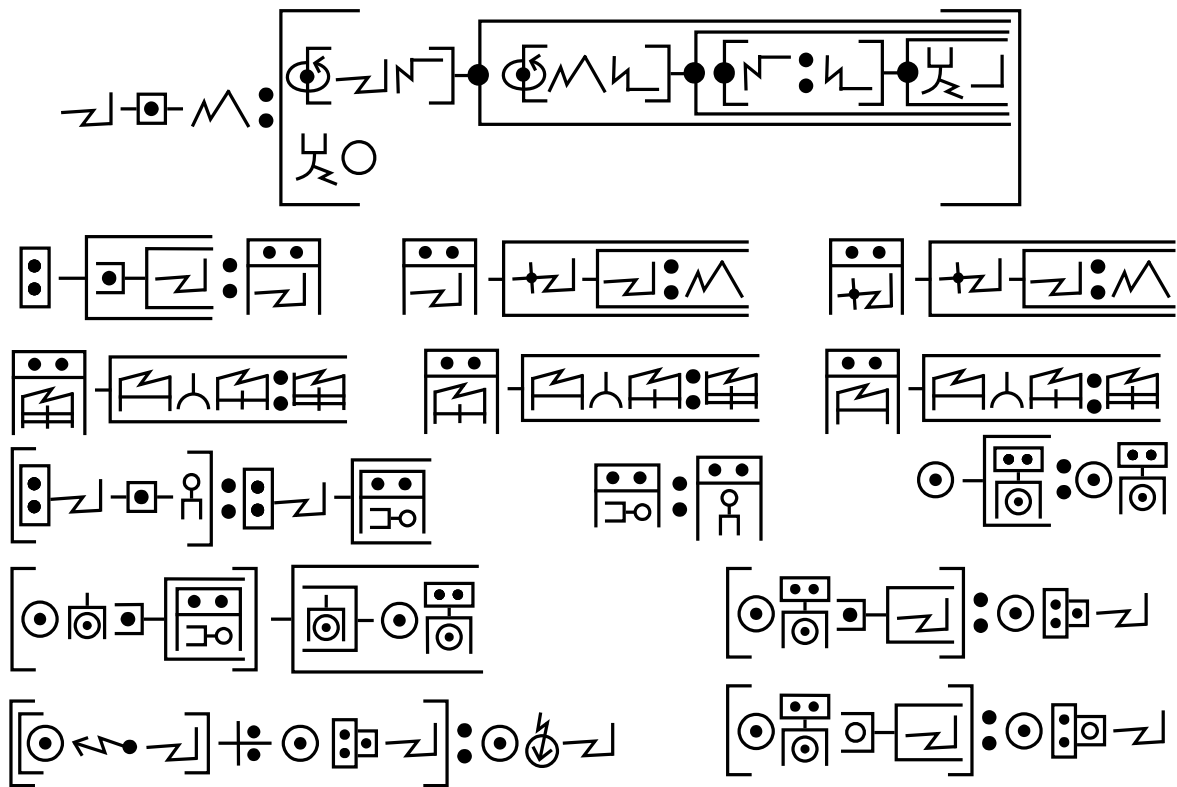

Top section : We create a function that checks if 2 arrays share any elements.
Second line : model sub(contains(a)) = model(a)   model(a) sub(time-a sub(a=b))
                 model(time-a) sub(time-a sub(a=b))
Third line : model (speed) sub(distance / time = speed)   model(time) sub(distance / time /speed)
                 model(distance) sub (distance / time = speed)
Fourth line :   (model-a share-element environment) = model-a sub (model(environment))
                 model(environment) = model(environment)
                 system sub(mind) = system mind
Fifth line : (sys subsystem contains model of environment) sub((subsystem) sub-of sys-mind)
                 (sys mind contains a) = sys aware of a
Sixth line: ((sys receive directional interaction from a) causes sys aware of a) = sys observe a
                 (sys mind not-contain a) = sys unaware of a

Conscious & Subconscious

Models in the mind can be conscious or subconscious, this just refers to whether the mind is itself aware of the model. If the mind contains a model, but the mind is not aware it contains the model then it is a subconscious model, if it is aware of it then it's a conscious model.

Overall consciousness of a mind will be generalized as "number of conscious models divided by number of subconscious models". I realize this is a great oversimplification, but any better value for this requires assigning some kind of formula for evaluating "model value and significance" (a rabbit hole I really don't want to dive down in this intro document) ,so it's the best we have for now. I think it's plausible to argue "significant models can be broken into more smaller models or can produce more derivative models than less significant models" so it's not too bad as as an abstracted approach.
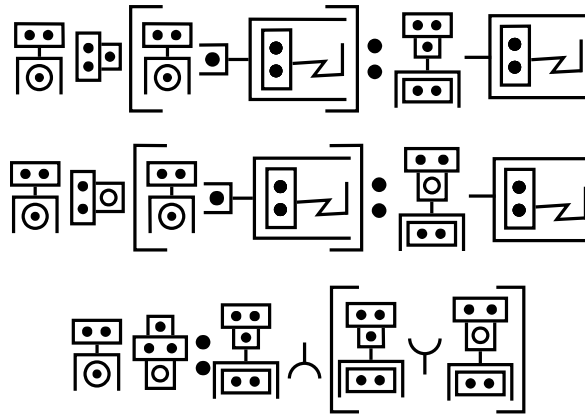
Top line :   mind aware (mind contains(model-a)) = conscious-mind sub (model-a)
Second line :   mind not-aware (mind contains (model-a)) = subconscious-mind sub(model-a)
Third line :   mind consciousness = conscious-mind / (conscious-mind + subconscious-mind)

Attraction

In physics attraction is always experienced by all participants, if A is attracted to B then B is also attracted to A. In more complex systems we often want to discuss "one sided attraction" (insert joke about relationships / love here).
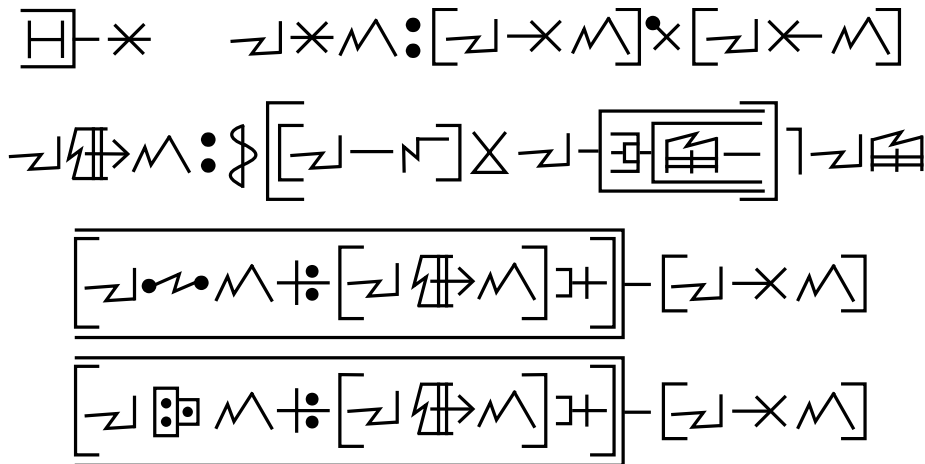

Line 1 : (bond) sub-of attraction   a attraction b = (a attracted-by b) AND (a attracts b)
Line 2 : a speed-towards b = sin((a line b) angle a sub(max(speed line))) * a speed
Line 3 : (a interact b causes (a speed-towards b) increase) sub-of (a attracted-by b)
Line 4 : (a aware of b causes (a speed-towards b) increase) sub-of (a attracted-by b)

Desires, Urges, Plans, Predictions & Probability

We will of course need to talk about the wants of minds and how they get it. We can break these into several easily and clearly defined categories.

    Desires – Conscious wants
    Urges – Subconscious wants
    Plans – Models for future action

To describe these methods well we first define the concept of predictions and probability. Probability can be messy to define, but physics gives us a great tool for doing this, particle half-life.
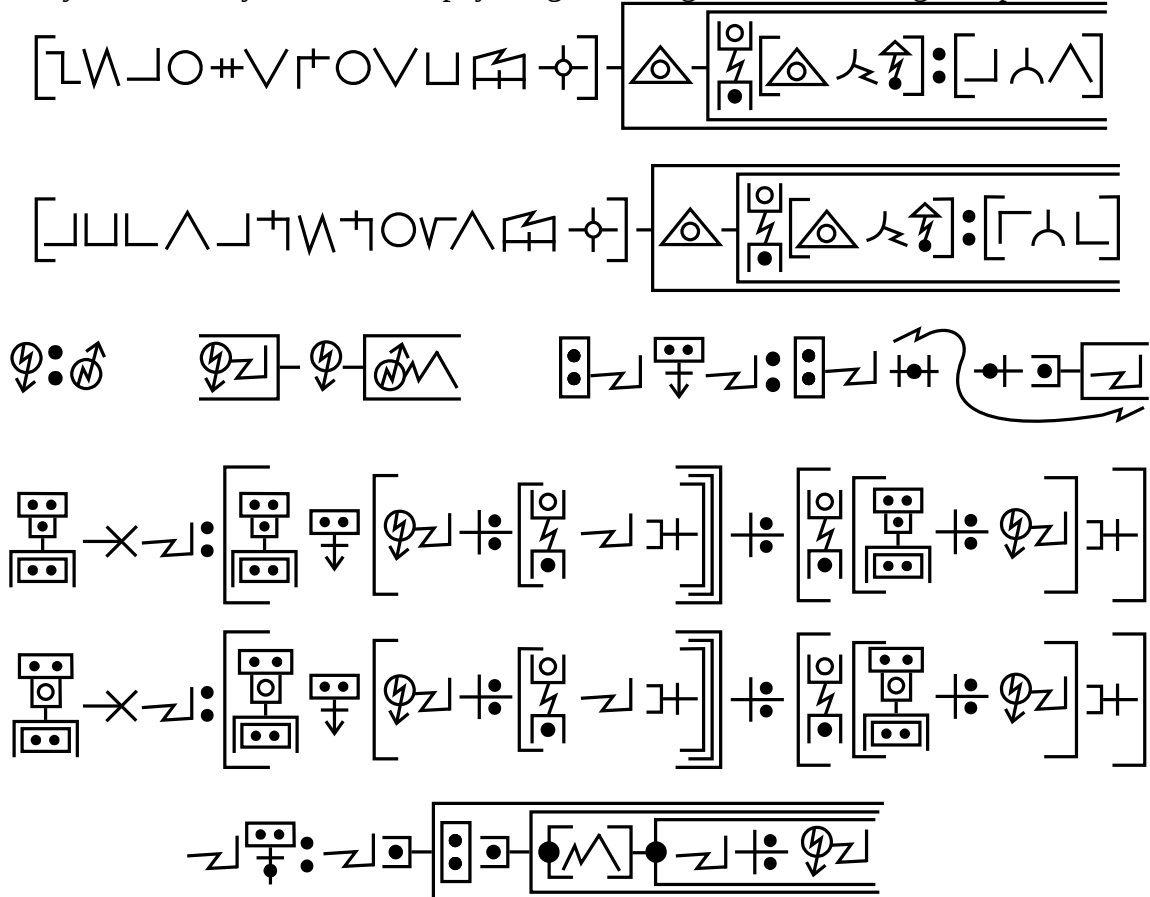
First line : (0xCA10F57059 time-unit period) sub(neutron sub(probability (neutron decay)=1/2))
Second line : (0x19421EAE0B2 time-unit period) sub(neutron sub(probability (neutron decay)=3/4))
        *"probability of neutron decay in 611 seconds=1/2, probability of neutron decay in 1222 seconds=3/4"
Third line :   alternate symbol for action   how to create action-variables
            model-a predict a = model-a state convert next contains (a)
Fourth line : conscious-mind desire a =
            (conscious-mind predict (action-a causes (probability a grow)))
            causes (probability (conscious mind causes action-a) grow)
Fourth line : subconscious-mind urge a =
            (subconscious-mind predict (action-a causes (probability a grow)))
            causes (probability (subconscious mind causes action-a) grow)
Fifth line : a has plan = a contains (model contains(if(b){a causes action-a}))

Notice that urges and desires are only dependent on whether the conscious mind or subconscious mind is working to cause the result. Every urge or desire increases the probability of the action occurring. When you feel an urge or desire the probability that you perform an action that you predict will provide the desired result increases.

Plans are just conditional models for future action, even if the condition is just "if now-time = x"

Learning, Forgetting & The Scientific Method

This section is drastically oversimplified. I am not entirely happy with it myself, we only define forgetting a model, we don't discuss knowing models are incorrect or applying and using a model vs. knowing a model, and we greatly oversimplify the scientific method. These oversimplifications and oversights are just because to deal with the issues properly requires a large volume of terms and long definitions. These matters can be addressed in a later add-on. The complexities of models can easily be an entire document on its own.

For now learning is whenever a mind gains a new model, and forgetting is when a mind loses a model (remember a model can also be pure information or data).

Evaluating a model and the scientific method are defined very abstractly for now. The main purpose of this document is to establish a base vocabulary for some very basic linguistic expression.

Here we also create a few new terms for "increases probability of" and "repulsion".
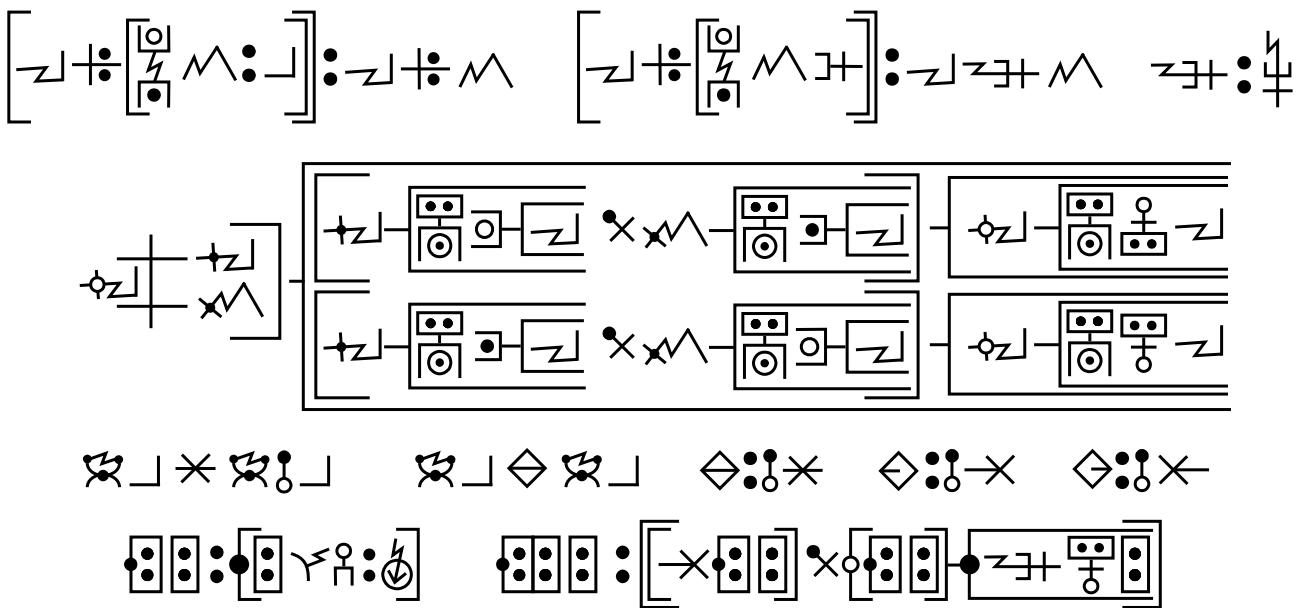
Top line :   (a causes (probability a = 1)) = a causes b
                (a causes (probability b increase)) = a increases probability of b
                increases probability of =   increases probability of   *just an alternate form of the symbol
Second section : a basic time-line with points and period labeled :
        Expressions in subset :
                (time-a sub(mind not-contain (a) AND time-b sub(mind contains(a)) = period-a sub(mind learns a))
                (time-a sub(mind contains (a) AND time-b sub(mind not-contain(a)) = period-a sub(mind forgets a))
Third line : electric charge 1 attract electric charge -1   electric charge 1 repel electric charge 1
             repel = invert attract   repelled-by = invert attracted-to   repels = invert attracts
Fourth line :   evaluate-model model = eval(model input environment = observation)
                scientific-method model = ((attracted to evaluate model))
                            AND if-not(evaluate-model){increase possibility forget model})

I do apologize for the superficiality of these definitions, but I want to restrict this document to a reasonable length. These grant just the bare minimum. For more formal description of these terms or other not included terms you can just create your own definitions or use long form expressions eg curiosity = "mind attracted to causing actions that cause observations that increase probability of learning", stubbornness = "mind repelled from learning models that predict another model is wrong", untestable-model "model that is impossible to test and produces no predictions".

Communication

Communication is an important type of interaction. Please remember that as with most high level concepts in Uscript, the English words we use to refer to our concepts are just "ID" values, the actual term is defined by the Uscript definition, not the English word.

Packet – something sent that is intended to be received.
Signal – a packet intended to cause a reaction
Message – a signal meant to convey information (the intended reaction is learning)
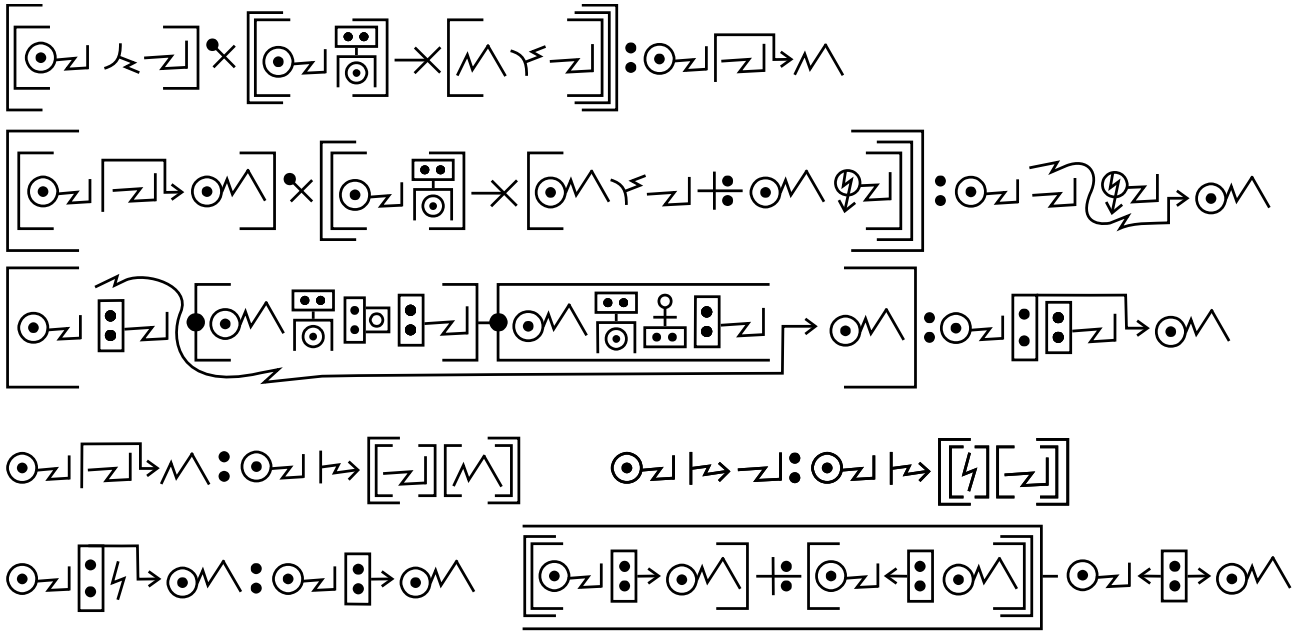
Line 1 : ((sys-a emit a) AND ((sys-a mind) wants (a absorb b))) = sys-a send(a) to b
Line 2 : ((sys-a send(a) to sys-b) AND ((sys-a mind) wants (sys-b absorb a causes sys-b action-a)))
          = sys-a signal(a | do | action-a) to sys-b
Line 3:(sys-a signal(model-a | do | if(sys-b mind not-aware model-a){sys-b mind learn model-a}) to sys-b))
          = sys-a message (model-a) to sys-b
Line 4 : sys-a send(a) to b = sys-a send array((a)(b))
        sys-a send-packet-to a = sys-a send ((undefined)(a))
Line 5 : sys-a send-message(undefined) to sys-b = sys-a send-message-to sys-b
        ((sys-a send-msg-to sys-b) causes (sys-a gets-sent-msg-by sys-b)) sub-of bilateral-comm

There is still plenty that can be added in this area like encoding systems(languages), mediums (sound, EM signal, writing), databases and storage formats, etc… Here we have just enough for some rudimentary conversation, this is another section that could easily become an entire extended database on its own.

All targeted system to system interaction can be classified easily into one of these 3. Everything is a packet, some packets are signals, and some signals are messages.

A broadcast is just when the intended receiver is "any of a group". We can also easily describe intercepting/receiving packets not intended for others.

If a system absorbs something that was was not intended for anyone, then it is not a packet, signal or message, if the system learns something from it that was just learning/inferring, communication requires conscious or subconscious intent in Uscript.

Pronouns, Instructions, and Questions

We want this to allow linguistic expression, and pronouns are important, at least me/us/you/we are. Singular and plural are also important. He/she/they however is not important, unlike "me" or "you" I can't just say "he/she" without there first being a context for who he is, since we would need to define that first eg. "system-a = he/she" or "context(he/she)=system-a", we might as well just call it "system-a" and skip all that mess.

There is also no gender, of course, because gender is a human construct, not universal, even on earth many life forms have no gender. Sexual reproduction can be defined, but is not necessary so I have omitted it for now. The tools needed to describe it are all available should one wish to do so.

Below we also create a "can/is able to" symbol as well as some symbols for "system with a mind", system without a mind" and "system capable of composing/sending messages". A system that is capable of sending a message must form a message which implies it is capable of imagining other minds and can image future actions, so this is a system with high levels of consciousness.
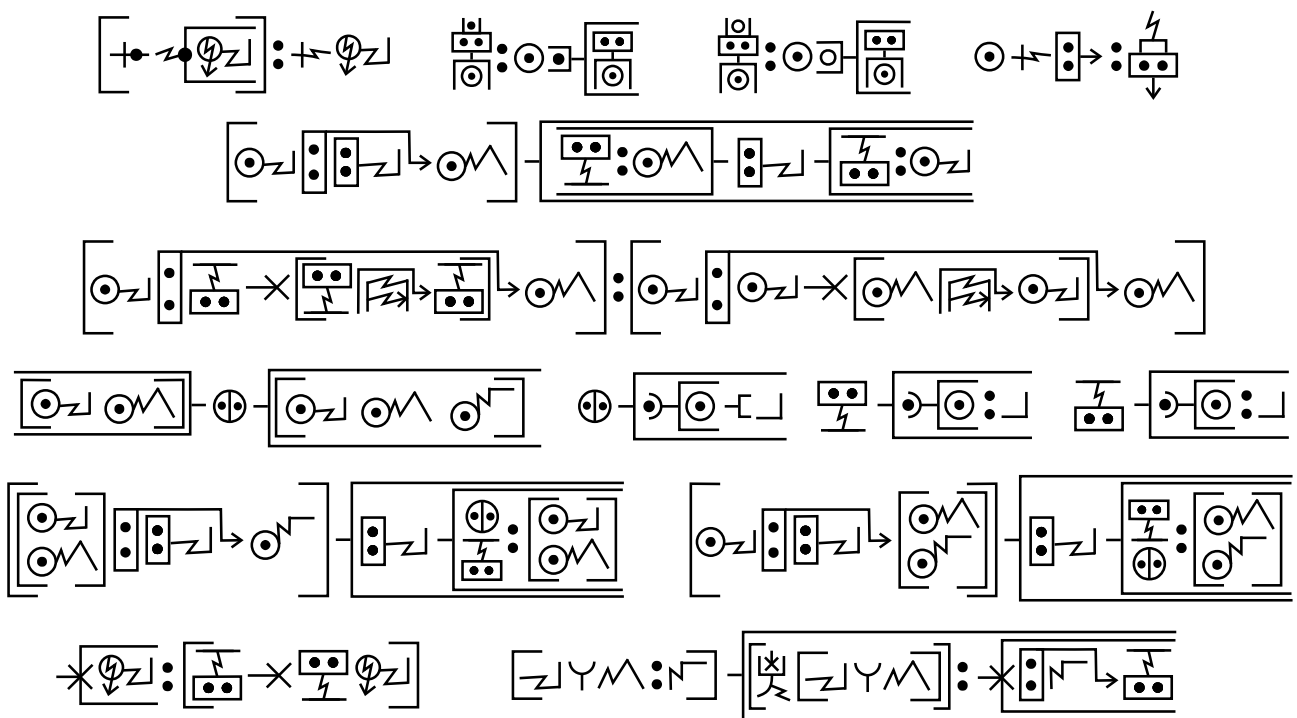
First line :   (future-state possible(action-a)) = can action-a
                     system-with-a-mind = system contains(mind)
                     system-without-a-mind = system not-contains(mind)
                     system can send-message = system-capable-of-language
Second line : (sys-a send-msg (msg-a) to sys-b) sub ((you = sys-b) sub-of msg-a sub (me = sys-a))
Third line : (sys-a send-msg(I want (you send (energy) to me)) to sys-b)
                     = (sys-a send-msg(sys-a want (sys-b send (energy) to sys-a)) to sys-b)
Fourth line :   ((sys-a sys-b)) sub-of group-of-systems sub (sys-a sys-b sys-c)
                     group sub (count(systems) > 1)   you sub (count (systems) = 1)
                     me sub (count (systems) = 1)
Fifth line   ((sys-a sys-b) send-msg(msg-a) to sys-c) sub (msg-a sub(we = (sys-a sys-b)))
                     (sys-a send-msg(msg-a) to (sys-b sys-c)) sub (msg-a sub(plural-you = (sys-b sys-c)))
Sixth line :   instruction (action-a) = (I want you action-a)
                     (a + b =c) sub(question(a+b) = instruction (send-message (c) to me))

With Question we didn't actually specify that we want them to calculate the answer, just that we want them to send us the answer.

Post-read FAQ

Q:We don't know neutrino masses accurately.
A:Yes, I am aware. I had not defined less than or greater than at that time so I just used equals to. The definition of < and > should be moved up to the top of the doc in the first maths section and neutrino masses should be mass<x not mass=x.

Q:Why no calculus? Why no field theory? Etc..
A:Because most adults have completely forgotten calculus. I don't want to drown readers in math that will turn them off. It is very easy to add it, but I was able to introduce more than enough physics concepts and define everything clearly without it. A secondary goal of Uscript is to inspire interest in science, so I try make it appeal to as broad an audience as possible and introduce interesting physics concepts in ways that don't overwhelm the average reader.

Q:Why does your style and structure change and vary throughout the document.
A:Firstly because I had notebooks of scribbles before writing this document but only in writing it did I formalize and standardize it properly. Secondly because there are many different ways of expressing things in Uscript and I want to give examples of various possible styles.

Q: I found a contradiction in Uscript structure/terms!! is this an error?
A:Possibly. I am far from perfect, but it's also possible you skipped something and/or didn't properly understand the definition. In higher levels later in the document I am assuming you understand the structures and terms laid out in the previous sections. Make sure you are reading the graphics, not the English text, the English text is just an aid, not the definition.

Q:May I use this in my project
A:Yes. You surely may. it's free for any use, even commercial. See next page.

Q:I found an error, how can I report it
A:I love bug reports. Send it to me at **bugs@dscript.org**

Q:Why no co-ordinates systems?
A:Because I didn't need them yet. Co-ordinates systems are used for specific descriptions and advanced maths, they require a fair amount of definitions to build them up but I was able to live without them so far so I didn't bother. They are of course needed eventually and can be added when desired. A few 1D and 2D examples are all you need to establish the structure for higher dimensional coordinate systems.

Q:What about space coordinates for our planet or astronomical bodies.
A:This is a complicated problem. There are no constant absolute reference points in the observable universe. The best we have come up with is using pulsars based on their frequency and distance, but even this has limits because over long enough period those pulsars will move, merge, die and their frequency will change. So there is no permanent universal solution available.

Q: Particle spin is not the same angular momentum, it's just a name given to a property of particles!
A:True, but the mathematical model for spin resembles the model for a spinning body, that is why they gave it that name. It doesn't matter if they are different things, the models are similar enough that it makes a good abstraction and analogy.

Resources / Links

Free for all to use, & modify
Uscript and all the graphics in the document are Creative commons.
Free to use in any project, commercial or non-commercial.
No Royalty, Fee, etc..

Here is a zip with the svg files (messy but it should all be there)
http://www.dscript.org/uscript_svg.zip

If you enjoy this script you may enjoy some of my other scripts.

Dscript 2D writing system : http://dscript.org/dscript.pdf
Dscript is a 2D writing system, it allows words to be writen as strings and characters that are still legible. It is not ideal for computer use, and in fact is very resistant to OCR (computer Optical Character Recognition).

Cscript Computer-Human Bi-freindly Script : http://dscript.org/cscript.pdf
Cscript is designed to be easy to OCR, easy to read and write, and have several layers of compression to allow for spatially dense writing.

Escript Electronic script for Low-res pixel display Script : http://dscript.org/escript.pdf
Escript is optimized for both handwriting and low resolution pixel display.

Chemical Calligraphy Basics : http://dscript.org/chem.pdf
Chemical Calligraphy Advanced / Artistic : http://dscript.org/chem2.pdf
Chemical calligraphy is a derivative of Dscript. It allows chemical structures to be drawn as symbols. The basics pdf introduces the basic principles, and the advanced pdf demonstrates how to use it as an artistic and mnemonic device to help when learning chemical structures and organic chemistry.

WireScript : http://dscript.org/wirescript.pdf
WireScript allows text to be written with wires in 2D and 3D. WireScript turns text into physical 3D art.

NailScript : http://dscript.org/nailscript.pdf
NailScript is a way to write text with a hammer and nails. By hammering nails into wood or other materials very durable and long lasting text can be written.

More Technology-Art and Constructed Scripts at http://www.dscript.org