Laegna Numbers

Examplified Manual

Denary

Signed

- O: -1; up; limit to -1 or -Zero
- *U: 0; middle (inwards); limit to 0 or Zero!*
- A: 1; down; limit to 1 or +Zero

Unsigned

- *U: 0; middle (inwards); limit to 0 or Zero!*
- O: 1; down; limit to 1 or Zero
- A: 2; up; limit to 2 or 2 Infinitesimals

Tenary

V: Upside down U

Unsigned

- I: -2; down; limit to -2 or -Exzero
- O: -1; up; limit to -1 or -Zero
- U: 0; middle (inwards); limit to 0 or Zero!
- A: 1; down; limit to 1 or +Zero
- E: 2; up; limit to 2 or +Exzero
- V: 2 · -2; outwards; limit to between 2 and -2 in Order; or Exzero!

Signed

- U: 0 · 5; inwards / upwards; limit to between -2 and 2 in Order; or Izzero!
- I: 1; up; limit to -1 or -Zero
- O: 2; down; limit to 1 or Zero
- A: 3; up; limit to 2 or 2 Infinitesimals
- E: 4; up; limit to -1 or -Zero
- V: 5 · 0; outwards / downwards; limit to between 2 and -2 in Order; or Exzero!

Decimal *Signed*

- 9A: Velocity=-V (imaginary receiver)
- 1: Velocity=I
- 2: Velocity=O
- 0A: Velocity=U
- 3: Velocity=A
- 4: Velocity=E
- 5: Acceleration=I
- 6: Acceleration=O
- 0B: Acceleration=U
- 7: Acceleration=A
- 8: Acceleration=E
- 9B: Acceleration=V

Decimal ***Unsigned***

- 0: Jump to 9 (Velocity=0); Value=Velocity
- 1: Velocity=1; Value=Velocity
- 2: Velocity=2; Value=Velocity
- 3: Velocity=3; Value=Velocity
- 4: Velocity=4; Value=Velocity
- 5: Acceleration=5; Value=Acceleration
- 6: Acceleration=6; Value=Acceleration
- 7: Acceleration=7; Value=Acceleration
- 8: Acceleration=8; Value=Acceleration
- 9: Jump to 0 (Acceleration=9); Value=Acceleration

Decet

X=X

- 1: Y=X; exZ=exZ
- 2: Y=Y
- 3: Y=Z

X=Y

- 4: Y=X
- 5: Y=Y; exY=exY
- 6: Y=Z

X=Z

- 7: Y=X
- 8: Y=Y
- 9: Y=Z

X=exX

- 0: exX=exX

Coordinate Systems

Unsigned Coordinate System

X=1

Y=2

Z=0

Signed Coordinate System

X=0

Y=1

Z=-1

Z can be seen as 2 (infinity crossover with E) in Signed Coordinate System; how could it be 3 in Unsigned System if 3 is not specifically an infinity around here, unless it’s Decimal Coordinate System, which is particularly not Laegna Number System! Z can be Y square very well, such as a value symmetric to YY or Y^Y, which as you see have the same number notation YY with symmetric digitalization, because position of Y in Y^Y is definitely before position of Y in it, respectively pointing to 2 Y’s.

Logex Ten

Logex is a Logic Machine and for it is Ten and Tensor given here:

Logex Tensor:

In brackets of Names: optional letters (implement full syntax of alternatives, such as "/"). Multiword names are allowed, which is IDE *linguistic modification* and AI *meaningful analysis* to proceed with that, foresee that and junk-analyze, clean and stylify the results, optionally or potentially by user confirmation, fix them.

Interface Tensor for Tensor(*s*):

- Input Portal: Map of Combinator*s* (iterate combination*s*)
    - Callback: automatic gradient backpropagation
- Receiver Map:
    - Input: *from* Input Portal
    - Custom: Integrator *receiving* Combinators
        - Integrator: *receiving* Combinators; *executing* Tensor Database

Logex Combinator:

Interface Combinator for Combinator(*s*):

- Integrate list of all combinations given, which yield some results
    - **Aggregator**: Give if the result thus combinator, in same but componentizable into two dimensions: the result is what is the logecs ponegate value of this combination, whether it's I, O, A or E, where zero (U or reverse-U) means it is not given in the list, or given in U-extended listing defined by user as Combinator table header and content item, where content is class and item is it's class listing.

    - **Processors**: Which classes are working to process this item, classes in instances in particular, but which can be seen in class item listing. Even if written in file, processors must remain as APIs containing enough code for files.

Logex Memory and Files

Links are either Memory or File Blocks or shared Internet items, including intranet and "*local*" community as chosen by AI from internet and local network; each File or Folder item is made of Blocks, which can *be* either *files* or *folders*, visible subitems to *exter* (outwards) files and folders.

File API or it's *driver* or *mother program* can provide visible (given the file is open or loaded as module or executed as program, if it has any given APIs); API can be included to *Blocks* or used as *Blocks*, given the Agile API it would be candidate for *second iteration*.

Memory or *File* Blocks disappear if:

- They are not given to Local Database, where they can hide their local content behind virtual items, protected database windows, API's and virtual files and folders; all this only inside their item – but user can link the item into any active listing any *Block* can have.
- They are not in memory listing of an active program, which is not obligued but it can close the active listings if closed, where calendar can disagree and keep some simplifications or AI memories.
- They are not *linked* to any *active item*, where even *folder*, *file* or other *block* can be *active* in this sense, files and folders belonging necessarily to block listing to be linked.
- Database item, which is activated by *block* itself, including if it's file or folder, now not italic as already defined as blocks and not necessarily added here.

- Memory of Program Listing can have active items linked, including their code, visibility settings and whether they require licences of any program or keep anything in program’s owners, such as copyright holders or stakeholders or head programmers of public GNU licence, who might also use other licences for some instances of use for fair access rights.

Databases are available to Blocks:

- Based on it’s interfaces, types and classes, block or it’s block can connect to databases, becoming visible item but hiding it’s private content.
- Web contains Blocks and web pages themselves contain Blocks.

Linking

Folders and Link Targets (basically folders) are available; Folders are containers of something, and possibly an item might by default disappear from it’s first place when folder is changed, but this can be changed; Link Target registers it to type, database, connection to another thing; to move from program’s memory to file system or to permanent database is to move into link target of this thing.

Anything can have multiple link targets or folders.

Program

Should appear in program listing, configurator listing, terminal command list etc., each being linked by some element of a program.

Otherwise normal file.

Block Structure

File has *mother programs,* which can give it it’s own resources such as having common web space for all it’s files. If it has several, they might not share those resources with one another.

- **Aggregator:** Something in terms of Laegna, which would be a type of “doemoticon”, such as Ponegate Value (Postion), complex Ponegate Value or Data Content of Ponegative number; in latter cases the search tree must finish with a single-valued ponegator of a few digits which can be seen with R=1 as single ponegative; the exact type of the ponegate might be given or a type understood by some programs might be given, where the type must be guessed by accelerators.
- **Properties:** Names and aliases, summaries, attachments (in a folder block).
- **Compression** and **indexing**, *cache mechanics*: file might allow to autocompress certain parts, others might be indexed or learnt by an AI, it might be registered by search engine for accessibility; the file keeps some record of this. Latency and speed, also potential use of resources such as two copies of itself when uncompressed can be considered.
    - Links might be used, where the is a link instead, for example a series of links referred by a web page might appear into this cache structure, enabling autocomplete or enabling the listing for programs; file might disappear leaving only a cache item to restore it; calculation result might fallback into it’s calculation score (energy effort is reported) especially if it’s lazy and not if the result is market as treasure to be protected by computer.
- **Tasks and Missions**: here, authoritative processors, web page service providers and users in they communities in internet, could see tasks and missions this piece is doing:
    - The block might have requests: it has data blocks, but those might have no information or low-quality information; it has requests such like fill this thing with this information or update the version, for example calculate to more digits.

    - It might serve promises, for example follow a promise to fill some information box to provide certain input to other user in the network.
    - They have ***karmic conditions***, such as certain calculation might have always provided something.
- **Generalizations and tags**:
    - **Tags might be assigned** (item is appearing in the tag list)
    - **Generalization to it's Interface, Class or Itself**: as text for AI or interlinking by implementing one API in api language of another, things might be generalized.
        - *Tasks* and *Missions* are **generalizable**: it can be identified, if they are partially the same (shared resource would have more resources); it can be identified, if the same AI student, which is autotuning itself, for example a single weight matrix, would consider to use common understanding template analyzing their slightly different APIs based on same models, and thus being able to inspire one based on content of another; this might provide shared context.
    - It might belong to chapter listing or file listing of collection, which links it into generalization as member of a book, movie collection, award set or for example an official art of a corporation, such as being a national city flag of some city.
- **Reports**: How much space it takes etc.
- **Headers**: Can be index of a text, list of files in a file container, row and column headers for a table, level list for a game; main block "Headers" (can be styled, but not removed) would give access to available content, but other containers might be available optionally, such as configuration, terminal commands etc – they are able for back-references, where a link might define a backlink, and when something is linked it's backlink would indicate this, or backlink such that if something is referring to it, it would backlink this; backlinks

can appear as link containers or file containers (generally called folders, especially if part of accessible folder tree).
  - Headers link to their content, most trivially text headers link to titles and content like paragraphs, a book collection might link to book files inside
- **Types**: each program can assign a link to type holder claiming that it's type condition is met, or issue that while having condition flag, it lacks the following criteria
- **Memory**: Contains data, such as images and text
- **Container**: Contains canvas, for example multi-page, sheet, Universe with invisible background, 3D space.
  - There can be multiple contexts: it has the desktop system's window system, program interface and template of paging system for it's actual content container.
  - The book might have pages, but it might have continuous space of web page, where it might have sidelink container for titles and content listing, but it might appear on blank pages where simplicity and elegance is added; additionally, the printer might mark the colors and filled areas as expensive resources, where graphical content is simplified.
- **Actors**: List of active items is given and they have *karma*:
  - Where did the solution of one block helped another.
  - Where the user did get help, for example it was in focus or used or given positive feedback.
  - Doemotions are marked.
  - Network of users is automatically sharing information and solving common ideas; in listings the interfaces appear which thus prefer certain solution types over another and include the standardization items, even to synchronize multiple groups the user belongs to. Databases, which are actively updated by the group, are also preferred by the group.
  - Actor has **resources** and **sustainability sources**: it might have advertising space, advertising target information access, synchronization of such with other programs, and

the services it has to sustain with advertising; for example it’s virtual block backed-up by cloud provider; it uses the general advertising driver of OS showing some allowed advertisements, and keeps up group belong of those information providers and their generation of revenue for the access providers; it also knows which users are actually buyers and get the real karma with advertising, or which ones might have been the influences based on their timing; or any anonymous source of information they have access too such as general summary of topic in Chat AI.

- **Interfaces and APIs**: Allow to access this with programs
    - Interfaces and APIs are *blocks*.
    - They can give keys and related identifiers to programs or other blocks, and make this visible or visible to some.
- **Identificator**: Active element, which makes sure different programs, containers and web pages are synchronously seeing this block as unique element or contained in unique element; it is able to use web page access or other named element access to prove with that element; such as cityzen registry in internet.
    - It’s able to see it’s address.
- **Identifier or Key**: public or protected accessor might appear, being able to protect itself in local computer, internet etc.
    - Contexts: Identifier holders might provide contexts, for example the container folder or file must necessarily have an access to identifier; for example AI is able to see an advertisement.
- **Templates** and **Type Holders**: Programs which provide type to this file, or it’s templates
- **Resources**: From *mother programs, other programs, websites* or *user, also the load balancers*, access to resources, such as memory, space, processor time, hardware or software or tools with API such as IOT could be accessed.
- **Virtualization**: By API’s and how the internal memory is accessed (active interfaces), block itself or it’s contained

blocks can be projected from a driver, not existing as mere data structure.
- **Databases**: providing APIs, Interfaces or logically belonging to class

Screens

- **Laegna Computer Central Screen** – In center, there is the big screen. Other screens are the same by the side they are connected with it, but otherwise smaller in the length which appears to outside.
    - **Time and Space**
      Time and Space have to do with organization of things in Time and Space, and while they are real time attractors, they can provide information of space and time in physics system, history and travel maps of pirate movie you watch, or any other organization of tempospatial space of Laegna.
        - Top Left: **Calendar** keeps undo lists, version control, memory, generalizes and connects it, for example to repeat an operation in different way. Variables, files, *blocks* in general contain concrete or general information about their history and the sources.
        - Top Right: **Map** is mapping the information and able to show the same or related variable spaces.
    - **Dynamic and Static Information**
        - Bottom Left: **Mathematical System** is able to detect variable content of words and numbers, mathematically analyze and provide mathematical solutions to everything focused or around or available in general form in the wider context:
            - It keeps track, where variables come from: for example, 3.14 might come from pi, the current state you copied might be result of your given calculation, not a mnemonics which can be served by this, and what you entered, again 3.14, might be pi as well or

        an operation with a number you got from computer yesterday.
    - Bottom Right: **Dictionary** and Page container has the API’s, Interfaces, Summaries and Definitions of the main block and the blocks of information contained, and where the user focus to which is the active and most probably dynamic item of each screen; with Dictionary, one can create the program listings – write active programs to this container, where moving from memory (working, visible) to hard drive (slowly processing over weeks) is not noticeable in terms of how the program is processing – both have some available processing power and space, and when computer is sleeping sometimes it’s not hibernated but dreaming: using a good time, when it’s not disturbing anybody, to process the items frequently for some time.
- **Windowing system**
  Windowing System contain components associated with applications (but not the documents), window interface and the operating system, but it’s less concerned with the details.
    - On Top there is **acne**, the **Window Control System**, which can contain “Laegna OS” and / or the program name in large, icons below this, and the chat input box above or below the icons where the user gives general commands in English or their native language to control the general environment, interact with the programs and run their tasks, including the ones hibernated on hard drive, which have an active component such as participiating in generalization of listing used by runtime, such like “3D graphics is allowed indeed by this inactive driver”.
        - User can set modes or tasks: “working”, “watching movies”, “connect two text editor windows translating from one to the other”.

    - On Middle there is the **main screen**, most actively organizing the running programs; for the Canvas (simple and elegant with no buttons, borders, window interfaces) or coupled Canvases (copying parts from one file to another, controlling this window with this chat), or movie or computer game again taking the whole screen. Other screens do not seem as “percents” of screen space, where they take actual space but make the border still very physical; they can still provide helpful things to a movie or game – upper screen it’s name in it’s own style and elegant fashion; history and map might be necessary for game or the movie; they can use toolbox or design box, for example 3D appearance control and change in some elements or selected parts of movie to watch – without extended information, such as added 5 min or introduction by author -; the game might have terminal or chat, dictionaries and math as well, where the movie might allow to understand things in terms of math or contain dictionary information like place listing of the movie.
    - On Bottom between math and dict there is Terminal, where there is the textual commander, log or any program able to use the terminal; it might contain graphical elements, but those wont disturb the flow, scrolling and availability of text, visibility of it’s controller and the absolute optimization coming from the speed of such format.
  - **Canvas Control System**
    You should associate the buttons, available program and control system with rather Canvas for the content of visible file, which could switch the programs even in real time, having multiple display API’s, or become on it’s own being view-only or having internal editor of some elements; while Canvas has it’s drivers linked to some programs, it’s the main visible element and not the Canvas

itself; for example, for Server the actual document is the list of processes it’s running rather than it’s abstract instance, which might be not so active, being a lower-level program not even to be accessed so fast, if this matters.

- **Left**: On the left side there is **display editor** (**Toolbox**, can be named **Edit** or **Editor**), constructor such as the Toolbox in visual ide for the programs; while the Application has registered certain buttons and tools, an add-on can add them; user for example installs specific effect into OS and not the program, appearing here for compatible API’s such as being a vector drawing [of given standard]; then, from here he drags the button into support system of this canvas, and user can apply the effect in this mode with this file, whether or not they have the program running. An AI is enough to synchronize these interfaces and their associated interaction, such as the program might set it’s navigation system such as scroll bars as optional and selectable, possibly by the AI, when user installs a navigation plugin for only this Canvas and not this Program.
- **Right**: The Design Tool (**Styler**, can be named **View** or **Viewer**), mainly Designing the Canvas appearance – but the program is not Styled on it’s own, but slave to the Canvas in this respect; for example if you watch a Christmas card or a Christmas movie, but the viewer and the movie player are expected to apply their Christmas theme or an AI might look for a Christmas theme for theme which is otherwise compatible with the current style so that the interaction is smooth.

Command Syntax, Paradigm and Running Order

Variable:

- From it’s introduction, where OA will set it’s preview or preliminary values visible in gray or distinct manner or with warning, to IE where the results wirst start to be safe to be used, later they are precise or in future they are the actual results for the estimation, where the feedback can be given.
- It can be **Tensor**: There is list of parameters, which have optimizer APIs as well for different use cases, and they can be linear digits of continuous number or form a discrete set of combinations, which have unique properties each and cannot be generalized into linear or exponential system. Tensor is very well ponegate: it can have input for **default**, which is used if it’s current content or content in one of the processing units is not better; it has access to dictionary and map and other elements of screen, either to their physical device or that aspect in local copy or fractal.
    - Tensor might be tense about where it belongs
- It can have **Strict Type**: either limitations, where some values are not allower, or boundaries, where all values must be in given scope; it has also the “rebels”, which are added but not following the type – for example unusual exceptions might be passed through; we are given that computer is not completely against exceptions with AI, such as fixing the HTML of a web page is most typical classic example, with it’s goods and the problems.
- It has **metaphysics** / **retaphysics**: if it appears similar to some other variables, common space for AI analysis appears, where R is able to contain both T’s or their generalization, being **do***mindful*.

Assigning and Commanding

Well commanding ..well, it’s so:
* Blocks: Blocks are run as infinite cycles anyway, but once the variable is known the cycle would pass.
* Subblocks: They know time in several directions, where they are after or before things in their parent block, but after it in the Y dimension of time, where they go inwards in fractal; where when running down, variables report back by circle through V, but when going inside this function call raport, what they report back to where variable was defined or first used in the same instance is the fractal compression of the information, where the last leave gives the content block, and the caller tree becomes the title system or structure of headers and subheaders, each having it’s feedback content available as immediate content of this chapter, not it’s subchapters with local-chapter-style counting also from below, of it’s callers.

Open System is *Open Minded*: I and E are not complete, but it’s forever sending it back.

Seamless to Hibernation: in Memory, the whole Container is created, and it’s activity is the ***Focus Mode*** of the content and it’s controller, such as the Database Research Element is active and focused when user is displaying the window, but focus is not zero when the program is not run. Anyway, when closed, where moved to hard drive from program memory means it was linked to be real file, but this can happen even automatically with drafts – in it’s virtual condition, it sees no variables or structures disappearing, but when it has more working memory at runtime, it needs to carefully move some elements to “cache”, which means to compress, to leave as a copy to server or to trash them to be deleted progressively of files used in the past; so it might not have more access once it’s trashed, it’s own link which might not be the link of garbage collector. In each case, some memory is left and the processing environment is systematically visiting the elements given their karma – how much they score in general progress of solutions or the user solutions, where one program might give useful data much earlier or some internet friend is to be used in sense of their computer generating information based on their files, where the content is anonymous such as selection of statistics of rabbits in the area.

Imperator

Imperator is Time – in sense of time, the programs run from backwards to forward.
Even logical variables follow this course of calculation.
Program lines are connected when future line uses information from past line.

Controller

Time is controlled by **Timer** or **Controller**:

- It contains necessary lines to contain in your controlled block, where controllers might be synchronized or communicating, for example the resources; it might say there must be switch block.
    - It contains if block, where one selection means one function would be called, with other selection other function would be called.
    - It flows along with this, and it checks that either functions are called where it's interface is registered for given object or environment of similar interfaces, or it might demand meposetive proof or any other criteria of this being the right timeline to align. It can receive hints about which ones would be followed.
- It contains the criteria you have to pass; for example OpenGL structure must be called in specific order, and so the Controller would associate the controller with the order for you, even if you could physically write them in random order

for example by syntax, not fitting the begins and ends immediately.

Logos

The Word is the Logic. *Associate the Magic with the Logical Paradigm – here, the mystical experiences might appear.*
When logical variable is assigned, change in the value might be acceleration, reflected in calendar and the map (to keep the same position in some instances of recognition by history, or to move in others where the social space might change for example).
The letter V is used in the general sense – it will give feedback in the other direction than the passage; with U and V one can control the progress forward and back, where passing U or V coordinates would shift the system up or down.

Commands: The Imperator

Logex machine works under imperative control, which means:

- Any order of commands and assignments or variable checks are ran in sequence, where the last assignment also would change, what is expected from an assignment and how the relations with other variables are considered; change in the same variable would also change it's coordinate space – what is expected; so if R would be lost compared to T, after the long run logical error might be encountered; while where you directly change the variable value, R would follow and the approach is the it's the right variable; you might change a conditional along, such as Time or Space coordinate or position in a list, a static Metaphor for spatial time (not towards *better*, but towards *around*, which is the *space* against *time*).
- Variables can change values.

Exclamation mark: !

Code is written in *sentences*. Either programming system or an AI would take care of the sentences, where first is looking for ordered sentences and the latter is looking for unordered sentences, which are not already considered mistakes, and is handling exceptions in normal-looking statements or mistakes in what looks like broken statement, but is not language joke for example. Jokes are perfectly allowed in code to test AI capabilities for long-ran scenarios.

Exclamation:
given that *Prolog* style is used, the same variable is assigned twice, where the Prolog variable name convention triggers:

- A copy of variable name in header treats this as a variable, and is given the small-caps style.
- A copy of variable name in header treats this as a constant, and is given the big-caps style.
- For the instance when prolog interpretation is turned off, it's used with proper case.
- Acronyms and one-letter variables are handled properly, so that the small-caps and big-caps word are left and look normal, such as different acronyms such as CIAKGB would normalize – CiaKgb (two famous acronyms I use, not that I would refer to any association with my programming within this or without, as Bruce Lee said – "without" is an interesting word explaining Laegna "ex"; without connotiations to Bruce Lee *indeed*).
- Otherwise, for example $ symbol might prefix the earning variables, which are to be altered; in case of simple assignments with no special meaning the clarification is not needed, for example "a = b" might assign b to a, not vice versa.

Logical statements: a => b, where b would follow from a, is imperatively seen as b = a, where value of a is assigned to b logically by given logical statement, which is thus treated as equal allowing one to do math and optimize variables by “<=>”, which in turing context understands it’s acceleration or decceleration and shows how it’s optimizing away from the value once the seeming optimizations have been ran.

Define property to Sinus function that given the value of A, in particular case sinus would be seven (an unseen, definitely rare case in actual world we know, but perhaps normal to somewhere, out there in the depths of fantasy realities such as Alice in Wonderland, where one cannot be sure, as Russell or Descartes would teach if considered in magic realms of mathematics, definitely out there one day along ones of those 1.(0)1 probabilities, where infinity of zeros might limit this out and destroy the last 1, as it’s equal to qualitative difference in 0.(9), where the missing digit “1” is constantly virtualizing, but then left out – in Laegna, indeed, combining such impossible number is not so trivial, because would it matter – series of “A” is guaranteed to flip off according to the rule of it being a limit value to U, and the rule is stated and summarized):
sin(A) = 7!

given that A = 7, sin(7) would now return 7, overriding the existing function qualities, even if the rest would work; using the approximation equality also allowed in =>, basis for = in opposite order.

Sin(a) = 7!

given that sin(7) already equals 7 as we executed the prev command (just imagine even if well it’s not good literal tradition to surprise like that), this would assign 7 to a, which is told to be changed in a way that Sin(a) would equal 7.

Nature of logical statement sets their value based on symbolic representation, and links back to original to make some miracles with comparison operator, which would not trace back based on general case even if it should, in some cases, and to answer the logical explanation; it’s a magic trick as it rather works by mathematical synchronicity than any solid rule – if the callback exists only on paper, the program might fail and the paper must be given, then, which is very important in case of Turing’s Halting Paradox, which is a little unfair to state that the relation must be *hidden*, given that in mathematics a trained policeman or a trained and fine-tuned AI with gradient backpropagation would be expected for that in the real world settings, but the initial system should be directly aware of some relations not in the testing set vs. training sets (in overall, why to not use U for training and test set verification, and not have the test group as Laegna does not want such loss in the information in the end, even if in the way it’s good to evaluate algorithmic parameters and find a balanced set, until when it might be “Latin” or involve the test groups).

Assertments: The Logos
The Logos eternally exists above Space and Time: whatever we do, the inevitable logic will meet us as stated in the mathematical description of Law of Karma.

Assertments:

- They are considered in order, where the output of the last is good input for the next.
- Where the value is explicitly changed, if the rest of the values change in relation to their control set, and the implications of it have also changed their values along with it’s considerations, and where the control variables such as calculation Time and Space, given as R values for this (where lists and dictionaries can also have one focused element, where we might need capitalization other time – can AI, with some basic considerations, separate between set-element distinction like R and r, where R might be mapped so that r

represents a combination sets of cases of belonging to R, and proving rules of found linear cases or their estimations).

- Logex feedback is generated if there is change in the value: it might be asserted that proper is to give a certain value back to the future value, such as R[global.time=50] = Position might state that at Turn 50 in MoM or Civ-like turn-based computer game, certain logical statement as referred by R must be at Position, thus a positive assertion or assurance, or expectation or trial is set for R to be positive at time unit 50. Sets, which are related to this time unit, would have variable r of property R always set to this particular time unit; then, while treating the Prolog names as generals and particulars, the Logex system would indeed refer to class description and the selection group for optimization or evolution, which randomly happens when trialing the possibilities with Zin "meditation" of computer or actually random chaos, *entropy* of all variables or intentional entropy – the R would be also a generalization of items in R, and based on r mapping of selecting an item, such as "UUVV", from dictionary, where V refers to digit values of the same variable in external set, where it's sometimes assigned something, assuming that unknown digits point linearly in infinity, and the unknown resolves based on other properties where this unknown is effectively invisible by philosophy; U is expected to be resolved later and the variable map balances and integrates two separate values, keeping this to be the database structure.
- Frequency would raise: the variable space itself, along with value, would shift up and down in frequency, which means the positive quality the variable has – this leaves trace as it's considered to be similar to imperative assignation, where time is the *imperative*, not to say *imperator*; which explain the imperative paradigm as pure *logically*, the *time* does *not* exist, but is a poor emulation. Time inside logic exists: T finities counting the operations, and R infinities measuring the qualitative progress of their execution, where the result of

calculation feeds back to solution set, where negative or no execution removes it's _meaning_ afterwards unless it's U, a trial for given set, where it's then Z value above treshold yet, but part of normal distribution by it's essence and intent – exact U value can be _intent_, you have only possibilities but by producing right balance and feedback, leaving each situation and word open in the right word, is the pastes basis of your past – "pastes" is combined like "antes", meaning it used to be past or *rather* that in past, it used to be present (*generally* the opportunity to leave it open). So, since it's an imperative, while variable is changing, so is it's space and would it change back, it would leave a trace that in some scale it's still moving to the future, even in some projective space we would shift back in time and remove it's consequence or support it's position consequence. The variable, at that frequency, is now having new value.

Proving and Probing

In addition to *Imperating* (the exclamation, change) and *Communicating* (the period, "logos" is the microworld or the world of words as atoms of communication; I associate with "lego" and "legos", which is sometimes my better word and associated with Women logecs, while the standard logecs is perfect for women being not very demanding on original throught, but "legodriad" is the long-term association in "eq" the emotional world, where one does "iq" operation in regards to infinity but in equally irrational basis to a thing such as matter, because emotions also do not have particulars; emotionally, things like magic, emotions, religion, ethics or sensitivity and compassion can be understood as they all work in the world of ideas, and their exact behavior depends on the extent the idea world can influence the reality, where the magic is *actual* only where the *real world* decides to follow that particular symbol or archetype or mental construct, thought in case of thought power T such as hermetics; mind map in case of mind power R such as religion or buddhism in particular, or social case E – E is also the plurarity like E, R, D and other infinities). So we are going to *communicate*.

sin(A) = 7.

This here, and unfortunately the previous commands now are not ran, where I also don't meet the stability no; but indeed the sin is now configured to stay reflecting with A with value of sinus function changing along with variable A, which is perhaps even less probable to happen with any values of sinus in the actual entity of the logic; but perhaps sometimes you don't want the base math value, but value in regards to your local coordinate system where 7 might point to actual value of the sinus of the functions, which are immutable or invariant to your coordinate system change of *symbolic system*, where otherwise for example you might have no values at all. For example you come so close to the numbers, and you say you have applied to *projective space* through a matrix.

Sin(a) = 7.

This time, Sinus function will remain constant, but where the actual value of Sin(a) would change, for example in terms of R multipliers such as Sin(a) = 3R; indeed there is something known about R now – in case R is rigid, we don't know what to do and some kind of error or confusion to be solved by an AI is necessary:

- Another time it's not a fairy tale or middle of chess game so I make up the rules now you did need in advance, to surprise you negatively but it's only a science: @a would define a symbolic map, where @a is a tensor whose type can be given, certain values assigned to be guaranteed by it's interface or implementation where an interface won't meet it's expectations even if probed by certantly conjecturing to assignation, which gives posetive and looks for negotive or

it’s “deeper cause” in this sense, distributed to logic and code irrationally and provoking the basis for mathematical synchronicity to arrive once it makes any sense at all in the logical and the real world, where it’s destined to pass trial and error if failing the conscious thought and logic – we should have gradient backpropagation here.
    - @a = 7
    - R = 2 + @a // notice while R is constant, but not in regards to @a which is symbol not value – when @a is changing, the implications change but if they contain symbol and not real value as part of it’s configuration, for example the variable is simply optimizing to have more digits not set to zero, for example starting with 70e10^700 000, where it takes time to pass anything into the quantum preciseness, or it might straightforwards rule it out by imprecision of available data, for example about the future involving an unknown event or several successful strategies but each in it’s own future set case, where we prepare rather for general life than our own specific story – perhaps, an enemy appears *in our equation*, and walks only in direction of our fear or worse, our visible fear – whatever we choose, we get not, which is the antiparadox or self-fulfilling prophecy.
- Calendar would register the feed-forward, assign a forward direction to this variable, associate them with past priorities associated with tensors (such as @a in our previous example); system of forward would generalize it’s feedback into a model, which can be used by source to get closer to it’s solution; being able to get it’s virtualized input as compared to the output, and assigning the correcture, or generating output and getting estimated input. Let’s call this probe-forward.
    - Probe-forward:
        - A tensor system, involving tensor and weight matrix, registers it’s combinators or data probes, actors

containing requests to grow in precision and accomplishment to actually verify the level of randomness, where non-random gives failures of failing certain expectations of opposite decision of belief; this returns from overfitting perhaps. The feedback system, given 20 static answer with 5% probability each, connects not to real answer but to probability map of it generating given answer, in current sequence, to the question; for example it might know it does not want to repeat the answer unless asked directly or having different question and when talking involves price, the fresh or repeated information in answer, unlike the triviality already understood and experienced intuitively at least; if it's not U or unknown, it would feed back – here, we can see where we could actually meet the value "0", for our original question containing "O" and "A" counterparts, a yielding solution; where "0" is told to be achieved in logical system by not doing any of addition, multiplication, subtraction or division, the code blocks would not fill this position even with space, which would separate it into words – then, the current digit is set to "None" and even not to " ", which is a space and means an infinitely-long separator of sequences, where sequences separated by spaces have their own infinities, and spaces represent then the actual infinities *between* them, a circular logical necessity to enable the existence of spaces, and it's practical usage that we rather need words and number counterparts of first-level matrix to have their infinite containers, than being simply separated by infinities in unknown manner by unknown theorem, where our numbers reside on axes not having this coincidence of improbability by definition unless we meet it in measurable scale.
Summary: tensor connects to it's *Reason*, or connects it's situation to it's *Cause* like recognizing to optimize

similar situation based on feedback; for example it can reach to similar input condition and alter the weight matrix with considerable consciousness of it’s actual unknown, and the process itself is balanced by metaweight matrix to properly conclude the feedback in future heuristics. For example, based on generalization, multiplication would actually strictify the exponential function where it ends with exponent function in a constant space.

- AI can fact-check or feed-back:
  - Once the *prober* has established a module of it’s probes to another AI matrix (I call tensors and weights combined a “matrix”, as it’s a multidimensional matrix by it’s properties in some projective scope), it can generate probable questions appearing in work of this AI at the receiver, or the code controller it has instead of AI or the symbiosis of user, code controller, dynamic feedback of scripting code, and the AI interacting with all of this.
  - When the source AI working with open questions of our self-referential AI, which has an open fact or open weight matrix or model structure of how to connect it’s information or a general description of it’s intelligently fitting model, even if it’s not in runtime but in file – the files, containing code and actors to promote this code into a hibernated function of non-zero degree experience in the thermodynamic system of code temperature, the temperature of the code is not zero whether it’s in memory and registered in program uptime, or whether it’s in file system, registered to OS process, AI analysis, tasks and open problems; simply by existing it would be able to process it’s situation and provide something somewhere; it has it’s local code and special access to it’s mother program, which is sure it’s it’s actual structure it’s able to provide with answers.

Summary: an AI is able to control the tensor systems, which want to provide their sources with information and keep it accurate and updated – it does so by having their potential request map, especially it’s history-based generalization, it’s interested in such aspects or distributions of past interests, and it feeds back the gradient backpropagation data to form a backpropagating model to improve the past results without direct connection of being the same model.

- **Generalizations**: Structures appear of specific actors and their associated data probes or solution set probes, and along with the variables and axes, these are slowly processed by the AI to generalize them into question and solution sets.
- **Inspirations**: an inspiration set can *hypnotize* the AI, where it *emulates realistically asking questions* dictated to it, and responds with weight matrix optimizations approaching it’s free-willed optimizations where it’s asking questions more or less randomly, by it’s own intent and temperature, either cold or hot. Feedback variable would then hit it’s target.
- ***AI optimization generalization***: As it’s working in *hibernated mode*, where even another not booted OS could still remain operating through it’s public API to file system with low-need optimizer in another OS, for example other Laegna OS version, the thing described here simply enables the optimizer: tensors and the feedforward-backpropagation cycle is not part of active model in memory, containing the tensors, layers, weight matrices, but those are contained randomly in filesystem and associated, and able to update their systems such as 24/7 news feed for program with some activity privileges having it’s source configurations distributed to random folders, but market as having real-time property such as 10 minutes of 1% activity per day,

    where it understands that it needs to balance it’s resource need and help others by creating intentional differences in calendars.

- Social AI
    - Interests and tasks are in files, and AI’s would form something cloudlike with systems based on trust; they might have initial lists such as trusted people in their party or company, to be in cycle of given private information. It might belong to world-wide communities, which, for example collect open questions in physical simulations necessary for you to balance your combinator into smaller number of trials, they would generalize the open questions by operation of Zen, finding a simpler web of balanced combinatorics while not trying to achieve the precision unable, and finally the network of computer would give generalized answer and feed it back to the source.
    - When answer is fed back, like in the last paragraph: backpropagation, based whether the answer finally optimizes the original questions, as referred by local system and it’s generalization of it’s actual questions, estimating the interest – it would feed forward into another weight matrix which is interested to know, how to base it’s correction on actual questions and problems of the user. Where real-projective comparison is used to consider the progress of a ponegative fact in temporal dimension, time where it would progress forward, as evolving, where one has a system of facts spread in time, such as questions of each user are equally interested not that one would be more interesting like future, and we would be bored of another user falling into the past in this space – rather, each user exists in the same time and now, each can pass further (“E”, position destance) or backwards (“I”, negotion destance) or they might not care (“V”, not reverse-U, which simply states

that U is not left but this is considered to be solved if all the other things are solved, V assuming the current digit is linear continuation of other digits, where the R is one digit larger, but T is not changing it’s value, where T here does not consider the *length* of the number). So, here, the communication efficiency is measuring the feedback in both directions, and deccelerated or accelerated variables in Z and Y dimensions can contain two persons in one variable, where the effects flows in two directions; while the multiplicity is 2, such as I or E, or intinity, such as I% and E%, where “%” is rather degree T than length and perhaps position R and perhaps A, the rather Logecal value vs. R the Real value, Real also comparing to Left in the same way – but Left, indeed, can be meant like “overleft”, but also in context of leaving things to other people, which can suddenly bring a lot of money, where R * L is much bigger than R + L or 0.5(R + L) where equal effort would bring equal income. Thus, D(R, L) or R [D] L, where operator D means their balance leads to multiplication, while their imbalance is comparative division.

Assertions and Fact Probes

Sin(a) = 7?

A fact probe: In local context, a fact appears to request whether Sin(a) is 7. When locally, this is requested, the answer comes faster as the question is active as long as it’s if conditions, select statements or appearance in active for loop or active history is considered. When other classes are involved, also in their context requests appear, such as Sin function itself is now aware that it’s function with this input and this expectation to output is running.

Try Catch

Sin(a) = 7?!

If this is *True*, the function fails – you don't want to find out this. Use not, "!" exclamation mark *before* Sin, where it ends the sentence only if preceded by connection to last character of sentence, or separated by space from operator, which would otherwise form another potential operator, and it's followed by space or whitespace, but not segmented whitespace where small symbol of ruler which looks like millimeters in measure tape would count from -1 to 5 segments, where it connects the words into separating them with infinities.

Prove

The actual proving system is not here used, because while programmers should prove their code, they do this with prover or they do it by intuition, which is very hard to implement and translate. Rather, they use prover like Coq or Isabelle, getting a proven line in advance; they can plan this into future, leave it to others, or trust in their intuitive proof and still be able to express it in strict way, for example that given sentence in program would participipiate in mission of their client, Catholic Church, to "follow the God", and where the program normally, based on user feedback, fails if the result of it's process is reported to fail to do so, even if it was good idea otherwise or a brilliant science investment, based on assuming it's now a Medievial Time where our clients got our program from the time machine – well how would we know, by stories of our Grandmother?

We are now happy if Sin(a) = 7!?

To be happy might be a tensor, which is actually now influencing the program to follow Sin(a) = 7 – I call this "truth-teller" paradox, where self-fullfilling prophecy is either good or bad: for example, influencing the function Sin(a) to be 7 could be achieved if people in all the implications and reasons as given to Math system using math screen (or virtual screen on windows of our laegna system), and conclude that philosophical and ethical unit of Sin(a) is R = 2R!, where we move the space of R and it's value so that the program, inside this block simultaneously after the statement, the X and Z time (in our relation to Y when Z occurs, where Z is progressive and Y becomes regressive when we remain looking it as Y once we are there; for real the same value into the past is Y, but once it's resolved and for all others that were, it would be X not Y as they make 0 effort where the Y appears, so if it won't appear 'cause of you the inventor, it's Zero or X, but if it won't appear because of them now, as their context itself is Y already done or planned, their value would be Z and with downwards accents that they went there, not that they are coming from there; with their own situation or estimation or understanding or mistake of their doings can be symbolized using accents below the line) where we see – if we really control the value, and the result is not included in the <u>reason</u> why we do that or implying the <u>reason</u> – it's *tensor* which seeks the given value or the whole situation to seek the value and decide on that is valuable, which means the ethics or meaning is deeper, involving the quality of the variable itself not the one given by situation, which is visible on IE axe of fact of given degree; the self-fulfilling prophecy is bad if the situation it produces, or the lack of the one left undetected or non-avoided, would provide the needs, and/or if the case of needing this and fullfilling the need, unlike the case of needing to do some art and then doing it, but like the case of stealing a bread because we have none can be excused sometimes to somebody, like people in your community normally controlling the kitchen, but you cannot excuse trashing your own bread to have excuse for such question, where simulants would do that – an emotional addiction or repetition of some sweet karmic condition, which can be seen as sensory plearures or addictive behaviour, following the secondary qualities you are supposed to enjoy rather for fact itself, where “the goal supports the means”, where when you go to trash your bread to be normally able to steal it, your *goal* is somehow directed by supporting the *means*, which is contra-logic in this known situation, even if some hard means could interfere with goals, such as good living of some, and cancel their definition and result – this is the ^R fact, where R with this rooftop function means that while the definition is failing, but it’s doing so on it’s own – your evil means can conflict with goals producing the means, while also the absolute and logical-basis rational means, such as *distance* is the basis of means such as *cars* and *walks*, and while we base on these means we would base on goal if we are there without having the distance at all, but while it’s a mean, then, in this chain of consequence or producing the mean, we Negate this contradiction of not needing such means as distance with our friends; we can see the distance is intentful and targeting us, not followed by their interaction and cooperation being more valuated and useful, or resolving the critical situation; critical situation itself produces the mean and is not a goal – it’s the *karmic relativity*, where the situation is the same, but your control, preintention and success of such, would determine whether it’s your mistake or evil plan, your success or your winning of the lottery, perhaps a theft if you gave them all the reasons and then your having the money is rather a mean than a goal, where you would deserve the money; R^ in any case means the mean comes back and not the goal, through solving the posedriad or roposetive, where these forms of ponegation directly point to this shape, locally you see your concrete action and it’s concrete mistake, and what you need to do – where roponegative is posegate, where I said women don’t need much of those redriads, and driadic fractals where it’s not even local – but from the situation being pronegative, prone to some error in advance, the actual practice of resolving the situation, given by authority or government, solved in our deep meditations, simulations and complex calculations or engineering and communicating with public; but rather to advice people we explain concretely what they need to do, and something in their life will be posetion thus negative, and we would negote it thus get the positive value of having it negotiated, where negotion needs past form – past, in laegna is Negation, and if it was in the past or appeared as a danger impossible, destroying rather another thing we could get from avoiding the avoidance, the Negotion appears in our action to Negote, to Negotiate about the thing, which is where it relates to real world. In “Legadriad” it affects that for example partnerships break, where men accept their roles and positions, women run trials which resemble driads in sense that having something in the air for society, where it resembles a court process giving a man some title in return to their long relationship, where he would rise or fall – while he is guilty if the woman was wrong and he did not succeed, the woman psychology of choosing a man and failing would hardly leave her guilty, where historically the man was just given and temporarly, the woman might still consider the suggestions pro and against given men, like suggestion of their mother, where the mother talks to president, parliament, workers and engineers, the pope and the officers, and rather finally it’s very hard to understand how they are guilty while it rather looks like an official process; men, rather, decided themselves about this partnership and if it did not work out, they are judged rather practically than as if they only represent the Mind doing just the Negative process without guilt – this is where, for women, the Negative would become Legadriad, which a little resembles a Legal Trial, where some kind of honour is measured and some kind of certificate or a trial is conducted finally by the Public and even by the Government, which is going to watch that they won’t meet for example – an intent rarely expressed by the men, who rather just walk away and express this intent personally, rather than telling you that they secretly need you despite calling the government each time you meet; where their government can simply be their parents, but finally we see men failing and succeeding for mystical reasons by women, where the women would fail or succeed directly depending on whether men hire them, work for them, or physically stop or help them –

which is not so complex and cryptic, but quite visible physically, so the more typical “posedriad” (women word opposed as “roposetion” would reflect the “women situation” with mens word) or “posetion” where men reason the “legadriad”. Actual operation behind the word “posedriad” is equal to coordinate transformation to form OPQR order from IOAE order, where it is not fit with it’s straight coordinate system of passing such states in order or visibly avoiding them in both life and number system with the same combination, not that they would go straight but that the *resulting situation* is straight, such as OORR which would somehow involve smootlhy solving the situation that problems appear *in the middle*, in the normal case and not somehow unexpectedly; where in the future case OORR would mean that you handled all the progress and properly foresaw the thing of the future, not pretending that you are solving it.

?! means you are asserting, but you leave the *reason* open as a question, which is for following:

- This is a programming language, not a prover. Prover can be automated to check all the proofs and to generate any proof necessary, but this is a separate time-consuming activity.
- Programmers are asked for both proof and practical solutions: by having the Conjectures, both proofs and practical solutions will be applied – where a program does not know it’s proofs in advance, it would know the proofs.

Assertions and Assumptions blocks:

Both to assert something, or to conjecture it, would set the “**proven**” flag, for example probing in advance, if you reach that line later it’s based on assumption that this thing is true.

Now, you could use them in combination:
* **Asserting** something means you check this automatically.

* Now, including the modules, which are **Assuming** it, they don’t have the “burden of proofs”: unmet assumptions won’t enter to your list as your program, at optimization level 0 it would need the proof and in optimization level 9 it would fail with error if condition is not met.

Functionality: the Functional Paradigm
We are also being functional now: the *cache system*, and the *AI actors*, together they go on with progressively and iteratively work on our information necessities, proofs for “?!” and “!?” and even estimations for “?”’s in our programs, not forgetting the other types of sentences; in case those are not influenced by output variables or unknown side-effects and conditions (or context), it’s visible that they can be used functionally, such as lazy evaluation and estimation of the results, Z and Y factors where direct computation is X, as it does not have angles in past and from the future, as estimated and later confirmed, rejected, reinforced by experience.

Symbols

Variables, when used with “@”, are considered symbols – for example, for “@a” is taken.

Consider:

# Set the range of a being between 0 and 9, where example range 1245 would be number between 12 and 45 using the infinity angle and approximity of half octaves, where digits position through frequency map from above to below.

a in 09.

# a is symbol, thus while a is changing or having different value in a block, which does not leak the values upwards into outer scopes;

now R, for any local or global condition of @a, in each situation equals a + 2.
R = @a + 2.

Consider that R is capital words, so you consider only all the values of R’s, not a selected r or iterator for selection when using a small-caps version. Each r, then, is now a + 2 given the local and useful value of “a”, or another “a” if exclamation mark was used to point the identifier’s name to another value, which can be different by value even if constant without breaking our past logic.

Let’s assume R is actually a list:

# Set value of R as dictionary of I’s which belong to range of possible values of a, as requested from it or calculated. Now, R can be accessed with anything of type of a, which was requested if “range” does not exist, trying to use it to determine the range if any logic is available.
R[I of range(@a)].

Given this situation, when R is assigned a value using the capital, the value might be assigned to @a to keep the program consistent.

Deduction vs. Implication

The program must find out:

- Given the variables a and b, it might turn out that when a > 7, b > 7. This is written either as “a>7 => b>7” or for our purpose equivalent “b>7 = a>7”. Each time one is true, another is true, and the *conclusion is given*. For the following, let’s assume one more scipture: “b>7 =: a>7”, where it does not follow that not a > 7 would mean not b > 7, but only the implication not the deduction.

- Given the variables a and b, it might turn out that when not a > 7, the assumption is *weaker* that b > 7, but it’s not implied that a <= 7. We use dot at our implication: “b>7 ·= a>7”.

Statistical logex machine

In addition to separate variables, Logex can watch the variable flows:

- Statistical assumption holds in long term:
    - Where a continuous flow of data or operations pass the given condition, while their local values are random, the general function can be decisive, for example random numbers between 0 and 9 can average to average of their extremes or close to this value, statistically separated, especially over time.
    - **Simulative training**: Simulative aspect of reflecting on statistical case would not provide direct answer to questions, but at the statistical checkpoint the accumulated value would be correct. There can be different mapping functions or hidden layers based on where the statistics would appear.
    - **Philosophical training**: Simulative machine for statistical result would ask to generate a list of possible answers, where missing an item would give more penalties than containing an unneccessary item. Laegna numbers could use knots, positive and negative digits in the same whole number, to track this in a way that both false positions and false nonnegotions would appear, where R and T would change courses in parallel, as with negotive / negative numbers they are not rearranged as R and T are turned to opposite direction at the same time, thus the resulting axe, where numbers are mapped to it in order, would not depend on “*contradictions*” or numbers growing smaller like Alice in Wonderland would rather *do*, but not at home but in dreams; having R=T but different angles, the result

would be numeric scale where the interaction of number space and coordinates in the system would be much more complex than mere opposites. The numbers would be closer to explain the complexity of inductive, deductive and random, where a chain of values at different frequencies determine the number, and they create complex multidimension – in Laegna, finally still equalling to proper and proportional number value.

- **Goal-based training**: where the means are holy and the goals are pract, the system would benefit more where the logical cycle ends. In the answer, a mathematical algorithm would have 10 versions for different cases, each running different scenario for 10 input configurations; we run 100 combinations. In the end, for our more generic theorem, as in normal Laegna philosophy of unifying religions and atheism, we would have a single result – intermediate calculations might not consider the variables properly, but a special score would follow the final result. Consider the case:
    - You have 10 steps of the answer, where in 5 steps intermediate calculations give their corrects and wrongs, and in 5 steps decisions in relation to their future outcomes are considered instead.
    - First 5 elements of solution go randomly correct and wrong whatever you do, so that their correctness is a random factor. This optimization would slowly converge to U, so that each of 5 elements, in the answer it would pass the 100 possible calculations, but each generalization for common case telling something, what holds anyway, is wrong – based on philosophical test it would at least contain the answer so it would score some. Last 5 elements suggest positive attitude, they consider that throughout the 5 points, while you did not gain anything in any specific case, in the end you had gained 5€ for sure gaining sometimes in the middle and

having a relation that if first 4 fail, fifth means you get it from your grandfather or sister's cousin instead of working or getting a free gift card. This 5€, along with understanding that it's not the end of the world would make the 5 last elements more or less correct, not failing in given direction.

- First 5 points are important and the last 5 points unimportant. By logex rules, getting the first 5 into statistical averaging, they would balance to UUUUU – given the tensors being optimized, whatever they do it's U, but the last 5 are either OOOOO or EEEEE – thus, despite being averaged out if the first elements yield between IIIII and EEEEE, they would define the result otherwise and you would be suggested to follow Zen and not complain for unknown on your way, live day by day and expect the great fortune at the end. This is what you need to do – not for it's being the best, but by life demands **something** from you, even if the something is not known; you have to make decisions and thus, when estimating with AI – you don't measure the quality of estimation in some random degree, but you compare it with your own estimation or action without, as the decision is definite Truth while having any basis is *matter of means*, and not matter of Goal where you would just reject the better-than-nothing solution, where nothing in this case is not good.
- It's better to define some statistical actors, who are watching the variable at higher frequency – where lower frequencies of local dependency are out of range of the number, sub-zero frequencies disappearing as if they were zeroed or equated with some activation function, the higher frequencies depend on longer cycles and get closer to E or R; when free variables are analyzed, possibly they map back to local variables, but now they contain U variables – this is what we search. The

optimization might now directly optimize, if the math is powerful, towards this value; or it might consider that certain sequences, where it has the context of past answers, is better than the other.

For example, in your long time plan:

- For every day you generate some data whether you would have free lunch or not.
- Your monetary balanced is affected by this, where if you don’t have free lunch you do spend some money, like 2.40€ for rice and each week some meat, living almost like Buddha in a sense.
- The local answers are mostly wrong, where you don’t know the exact days where the free people serve free food, or whether at your restaurant at lower floor there is some food left by your friend, who owns the place and knows you won’t pay anyway except for Christmas and New Year Eve, when you always come with your grandmother.
- For 5 months, each time the answer is correct: you have 200€.
- If the days would count well, the 200€ would be 40€ wrong, but the precise knowledge of days would make you accept that as the 40$ would not provide quality, but little difference in quantity in what you want to buy, for example some nails but you can basically do with less nails and you would have the chairs, where some random people missing a chair coming sometimes would somehow compensate that, and the complaints come even with more chairs as there could be more random people. But now the 5 months lunch days are completely random and average to U, whereas the 200€ in the end would depend on your decisions you follow every day, for example you could live with your grandmother, but the free restaurant would be not there and you would support the kitchen with little money every day or two times per week. So, as the more important digits can not be optimized,

the last digit would matter solely, and it would slowly optimize.

Given the right system of U’s other letters and dependence on unknown factors, where some other program can upright generate the unknowns to create a basic framework of estimations and train the algorithm, having “U” used as unknown and not zero would optimize it – use “!” for zero, but for U it would result in U! (“!” here does not end the sentence, but in my non-integrated general theory of languages I use it here as limit of the digit sequence), which means that the ultimate logical value is clearly unknown; for V, the upside-down U you would not expect the answer for this, but calculate how to survive the both cases, for U you would expect to somehow resolve each of the consequences and consider it; for ô-ôômega you would define you never know in context of your theory or applicance, and you would optimize the interaction of the results, for example some side-effects, where the actual truth is not known, might interact with aspects of truth in some way; for example you don’t know whether stones have soul – but you can know they are quite natural in human world and not punishing you for not considering their emotions, at least in any way you would understand for sure; so you would have some knowledge somewhat affected by having a stone – for example, stones on alien planet would outright upset you, for example with noise or tendency to look like s*it, and you would assume that it relates to having a bad soul, if they would do it really with their souls and not be somehow born like this without being guilty – with actual stones, usually you don’t find the actual benefit from such assumption, so the “soul” in the souls would be something different than what you would expect from humans if they sit there because their soul called them to watch the moon; rather the open view to moon is not the determinant of position of Stones and they travel for other reasons, we could conclude measuring the stones – we have a lot of logic now about their Souls, the relations, without even knowing if they have one! Those assumptions might be stronger than only the assumption they have not.

Measuring benefit and probability

Measuring and calculating the possibilities makes the result stronger, and introducing unknowns where we can avoid them makes it weaker – from simulation, where the AI is hidden some knowledge intentionally, some unintentionally where it happens anyway, and given some knowledge, it’s designed to often have the same resulting score in each case, but out *non-knowing* is stronger if we still collected the facts, and here “V” is replaced with “W”, or “U” is replaced with “V”, with the metaresearch on different cases, where the result is sometimes not optimized to be zero, the “U” value would replace with stronger “V” without any result, where we rather checked we use other digits as we don’t have the right digits, than relying to other digits without noticing.

Thus, we do Zin and Yen: we measure good and bad outlooks and accord them to reality; instead of only Zen: combining with combinations we have (we use Zen in two meanings: another one is using balanced set of combinations, but while we do this intentlessly, it’s the combinatorics, which comes).

> You would doubt if this is the philosophy of Zen: you are told to live in the present, and when I seem to break that, I don’t break the case that this means practical resolution and not the past conflicts and future certanty or good outcome, where you cannot guarantee them; rather, you don’t reject those solutions and they form a part of your meditation.
> **Computer**: is *active*, when it outright solves your things and interacts; AI computer has a lot of open problems and tasks: it’s having dreamless sleep if it saves electricity and avoids noise or generation of heat; it’s meditating or dreaming if it uses the free time in different ways to resolve this situation

and solve the things; it's meditating if it tries to find more general solutions, deeper reasons, and balanced views which fit the whole database and set of goals – the meditation is more like Zin, if it is idle but it works on particulars, such as certain databases; when it's interacting with user it's often Sin (Sinus) in sense that the particular program is used to solve particulars; if they are working on highest level of generalization, they can be said creative, spiritual or ethical depending on the means and goal of this.

So, how we measure unknown:

- Measuring the generated answers based on the long-term statistics, not immediate feedback, we can see how scenarios map – whatever it does in the middle, would be similar to hidden layers. With this task given, the statistics gets better by measuring the ends, in case the random stories generate results averaging to such end, which might be false in case of heavy butterfly effect.
- Analyzing it's output in relation to leaving open, and closing things, for example generating a list of answers, if missing the right answer or having the other answer is minus, especially when cumulated together, while not having the other answer or having the right answer is good – each aspect is on separate digits and the digits have some logic, for example measuring the difference of **lacking the wrong answer with cost of lacking the right answer**.
    - We avoid generating frequencer of 20 answers and choosing the most probable. Rather, we try to accept U's and generate the version, where those are correct, so that 0.2 probability case really appears with such probability etc.
    - The frequencer does not have to be in accord with the best answer, but for example having 1 or l being probable (number one or letter L), the answer would be "1 or l", not the slightly more probable letter.

Function definitions

Program is being measured in sense of functional programming: the program logic is measured about whether the output of given variable affects the input of the next one (the Logecs straightforward connection); in case the input is given to user and the feedback is received (a physical, material connection), or if the user could have the same value and use the result (a metaphysical, logical connection) – the AI or logical system would notice if such connection exists, or does not exist, based on it’s variables; or whether it does not know.

Optimization Levels

Level -9:

- On this level, all the given operations are processed in advance by the operating system, and in compile time, given that you do not change the code behind, the function and variable and any relation would have given it’s output into fixed, closed tables and constant and data files or other blocks, where the compiler would not do these possibly expensive operations (unlike Level 0, minus level can be used in conjunction assigning several levels, minus to lower and plus to higher line where you need to do this in lines). In Latin script you say -9, where you can assume “9” is capital letter, and lower 1 would be small-caps and refer to -9, as in Laegna from 1 to 9 we would run <u>in same direction</u>, where it’s visible how the decimals are opposing their minus values as in Latin Logic, not growing as in Laegna’s R=T or Hermetic principles of Polarity, where hate and love are not “opposites” and we don’t have “Knowledge of Good and Bad”, but instead they are different frequencies of the same thing.

- Result: an index, a hash map, different kinds of datasets for optimization purposes, which might reduce levels in your level 0 operations; if they are progressive, yielding some truth when some digits are known, and some certanty when million digits are gone but in thousand years, automatic recompilation and recheck would happen and you would include the quality measure in your program and compilation summaries; so you would make some progress even after your death, which is like reincarnation, not like Louis XIV who did not care at all, but cared of the opposite! So you could be better man than him.

Level 0:
- Variable, function, condition in regards of the operation, might specify the need for Level 0.
- Constraints might keep it in “*Tunnel of Linearity*”: Blocks in normal static language, where we are outside the paradigm of being “static” or “dynamic”.

Proofs in Action

Imagine the following:

```
for indefinite loop {
A = select random -1 or 1.
B = random number
C = B
B = abs(B)
yield whether B == A?
}
```

After some calculations for A and B, it does not known the intermediate results, but it would reach B == A as having statistical probability, which is worse than being always false in terms of strict truth, where you cannot decide anything neither true or false in general case.

Imagine the function definition:

```
Car = new Caller().
({
if (free) {
IOT.call(Car).
Throw Exception().
}
Car.succeed().
} as fun)() = {}
```

Here, with this kind of thing we would find the theorem perhaps for random case: “free” means, the if is executed freely if it fits the template in Car or Caller; as Car is Caller and not Callable, and calling is in interaction with succeed, having effect on it so necessarily being seen as they are having side effects if not considered together (important in general case, where functions can be defined on any sequences where the past lines are influencing the future lines, but if not the interaction is left free and not considered so carefully, only by AI whether it exists without, like user activity to introduce logic between them, or correlation based on untrivial fact). Without the if, “call” would not be measured by our function, but the potential check itself makes it important – now we *definitely measure that it’s not there before “succeed”, as it manages to give an exception and in given case, “Car” would fail with “being a caller not callable”!

Laegna Computational Language

Now, we have the description of the Laegna Language quite clear – for example, if my Laegna Programming Environment would be different than yours, and I might have inventions or solutions, while yours would be Laegna Environment, it would need your inventions and solutions; you would compete in a normal market, not implement it completely from scratch in terms of preparation, implementation and use. 50 pages is a lot and now we return to implementation of Laegna numbers.

Implementation of Laegna Computational Language

Prerequisite: We live in times of an AI, and thus we are interested in training a Code Assistance in advance; thus instead of implementing a single language, we target to have generators of Code, Output, task description of this code and whether the given example fulfills the task and why. We use different dialects to generalize to the idea.

Domain Knowledge: Laegna Computational Language is not a single OS, Programming Language or a Dialect, or even Programming Environment. Rather it's the language behind, which we use to think in Laegna Programming.

Number System: The 1st iteration, which already gives us something useful, is to implement the generation of questions, answers and snippets of knowledge for documentation, which regard to Laegna Numbers in particular precision and not the general knowledge. Number types, definitions, operations and values are presented and asking questions, something meaningful is expected from the answer.

General Design of Template-like Knowledge Base: Considerations of programming language are given, and the contradictions like "!" is good for exclamation mark to end the sentence, but also to get the numbers into periods. We work out metaparameters of template: metaparameters describe different versions of Language, dialects or languages in terms of Laegna General Paradigm of Programming. For each, examples to implement particular piece of parser, either in framework which allows to integrate with other pieces, or as a specific string processing task, which would only find this and perhaps some other constructs given the parameters. We don't know any single "ideal" for Laegna Programming Language, but ideally we have the Code Assistance to program Artificial Intelligences and Laegna Languages, where AI able to program another AI is the general case, perhaps the propagation ability for 7 traits of life, where in future it would be more complicated to find a concrete definition of Life and rather a Turing Test of computer capable of protecting itself, rebuilding the material solution, another computer, responding to stimulus and implementing more AI systems to answer the changing needs of the Environment; still we need some particular definition of Life – but we cannot fall into only defining "emotions" or "cognition" as we cannot physically access them nor differentiate by basic logic in logical environment; meanwhile, we can use dolife and telife connections and favour, as life, anything which supports life or resolves the physical contradiction of existence, also a big part of our existential philosophy – what it means to eternally exist in the material world, as a species, and to support our personal material existence and fluid continuation of our cause, the Philosopher's Stone of Laegna?

Laegna Number System for AI Training

Where we have multitude of bases for AI system, we now build something robust – if we train with this strict representation, where Laegna contains many innovative or inspirational, poetic or metaphysical aspects, where your special conditions of your own infinities are somehow applied; we are interested in implementing it on basis of having the basic training source for more strict logic you are working in accordance with, and which behaves like a normalized system – in terms of this basic system, you have some framework to explain your specific solutions against their strict mathematical meaning.

Parameters of Laegna Numbers

Digit dimensionality:

- Dimensionality = Real. This is one-dimensional Real Number Digit.
- Dimensionality = Complex. This is the two-dimensional Complex Number Digit.

Space dimensionality (bonus for implementing Laegna Systems):

This is a detail for future implementations or advancements.

We are in the first iteration here, where we create the robust combinator for Laegna basics.

- Two dimensional matrix of lines and digit positions solves, if the dimensionality of space is raised in accordance with size of the matrix, and where multipliers smaller than 1 would not keep the proportions, it would give perfect results with whole numbers or given that sub-zero space is growing and not shrinking, with more content in case of higher dimensionality and not merely more space for the content, which is mathematically wrong in case of comparing the spaces like cm^2 based on their simple properties – in Laegna number system, for any operation, for example 0.5 squared gives 0.25, but you need workarounds with real life situations – for example, consider the information content and number of digits needed for this range, and the case that two-dimensional 0.5 is larger and not smaller than one-dimensional 0.5; Laegna proportions use the operations often in accordance, such as having the unitary dimensionality and knowledge of the real size relations, or using the R property of Laegna numbers that numbers with more digits are larger, on both sides of the whole number and subwhole number separator of positional numbers guaranteeing that when you multiply numbers smaller than 1, the R would grow according to their dimension and you assume that while their face values are equal, they now do a longer sequence of influencing the number positively. Similarly, the dimensionality of a number means: in proportion to each digit, digits it has upwards and frontwards are equivalent even if they differ in power, and backwards and downwards digits would flow with the same speed each (in some cases the finite position X would be 1-based, where the infinite of Y would be 2-based, and the numbers would map with using positional relations in a way that each digit is in separate space and each number of same size of matrix, what you can write, would be different based on using A's and E's in it's position separately – given it's size, one would account for finite and infinite dimensions being separate dimensions even if having equal value, and solve real-life calculations). We can have any-dimensional matrix. Simplified reading of the matrix simply continues to infinity in each direction and averages the numbers based on X and Y position being the same – we follow this in what comes, if any, since the calculated number size and correlations would be correct as much as you can understand in one dimension, and we use the resulting numbers as face values when reducing dimensionality – an important operation. Alternatively the digits can be added, where each direction has power (compensate this with now not raising the dimensionality where the unit is cm with adding and cm^2 with multiplication, these cases yielding realistic results in simplified case).

- Knots could be solved with space complexity, as the next digit at it's negative values with last digit with positive values repeats the behavior of frequencies – for example number OE, given the base-2 system which does not have E directly, E would move to previous digit and became A, where it adds 1 to O which is -1; the result would be U.
- Matrices of space-separated numbers (where space is not referring to final U): take each number, it's length is it's power and it's size is it's value – statistically, take the average with power relating to statistical probability in a way that the influence of these numbers is weak.

Now let's enter the dimensionality:

We don't need the Real or Complex, but we state the SpatialDimensionality:

- **DimSpace**: Spatial dimensionality is 1 for only real digits, 2 for complex and real digits, where only complex digits would not give us proper numbers – we might distill them from resulting numbers in additional operation to have the core very simple.
- **DimWord**: Word dimensionality 1 means that we only have digit positions and their unknowns as spaces, numbers will be written without a specific notation. Word dimensionality 1 means that the number is separated into words, and if words contain spaces inside, not separating. For words themselves, if they contain spaces, let's always identify them with "{" and "}".
- **DimLines**: Matrix dimensionality means the number is on separate lines, 1 or 2 mean one line or several lines, where bigger number might express more complex dimension of numbers. We might have some variables to assume certain, small numbers of lines or accept any.

Dimension

Dimension is measured as follows:

- r: Small R maps the coordinate system distortion and other point size manipulations of space into R, and the digit space we use to represent it into T; in logex value r would be actual space direction, and r' the r prim would be the projected direction or how many digits we use. We use r in our context to allow certain number of digits.
- f: This is the virtual space of digits – each digit outside r is rather influencing the previous digit in sequence. This might add to number power, so that the resulting digit is more influencial with two values approaching the same direction, or it might not in which case the overleft simply affects the number; those two methods should give same result in comparison for most purposes if the implementation happens to be right.

Number Dimension

Numbers have the following dimensions:

- **2 or 4 is the digit dimension**: for example, base OA is 2, base IOAE is 4, Matrix is 4 given with 1 digit and decimal system represents 4 when using 2 digits, one between 14 and another between 48.
- **Frequential dimension**: Use 0 if U and V are not accepted, 1 if U and V are accepted for base-4 and U alone for base-2, which needs two digits to make representations I, E and V possible in your description.

Representation:

- Digits=1: The single-digit representation, where decimals give 1-8 at specific digit, one-dimensional numbers fit well.
- Digits=2: Each digit is represented by R and T, R contains the Complex and T contains the Real component of a number.

Base System:

- Laegna: Use letters for numbers.
- Latin: Use numbers for numbers.

Signedness:

- Sign=2: Use only positive numbers.
- Sign=1: Use positive and negative numbers.

Capitalization (affects the number order):

- Polar: use minus and plus, with minus count the numbers from up to down. Decimal system in our case does not support negative polarization – we lack the font, so I leave it for now and have a simplistic implementation.
- Linear: numbers read always in the same direction, both plus and minus.
- Caps=1: We use capitalization for extended range of the numbers, where + and – when used will reverse the digit position.
- Caps=2: We use capitalization for plus and minus numbers, where the small caps digit is minus. We don’t use + and -.
- Caps=3: We use capitalization for digits after the dot / comma, and if Caps=3 plus and minus might be used, but not the dot / comma, where without Caps=3 dot or comma is used to separate whole number and the rational part.
- If we do not use negative numbers, “+” is also not used to show that the number is positive.

Precision:

- Instead of using discrete numbers, we can have float or bigint for each digit, containing the precision.

Full Number System

Dens:

The last digit of Laegna Number, based on 2 or 4, can be divided into Dens. 2, normally, could form something like OO or AA, but not OA or AO; despite this, for complex R and T we would get differing solutions; most often base-2 itself rather resembles Dens, but in a hidden form we can know more precise values of it. We are more interested in base 4 and we rather divide it into two Dens based on it's content:

- Dens, generally, divide the number digits into two binary components; most trivially, for IOAE, we know that IO is local or global wrong where AE is local or global correct, here R knows whether it's local or global and T knows whether it's wrong or correct; alternatively, being wrong would switch the value and where one den, the R knows whether the solution was initially correct and later accepted, or initially wrong and later corrected, where the T knows whether the correct solution was to be correct or to be wrong in the first case – we we realistic and chose one of the two. Basically, we can consider each R and T combination – each two-bit combination to contain complete information about four-based digit – and then see whether they can be solved by other combinations (while binary has number of combinators or truth value tables – as much as 16 with two and 4 with one input, 20 in total for basic combinations – we can emulate this with only not, and and or (Bertrand Russell has analyzed this), getting the trivial three of programmers; with all these combinations, much like choosing not, and and or from 20 combinations – 2 from 16 and additional 1 from the 4 combinations; for example most trivially we don't need the combination, which is opposite of not and simply leaves the one input parameter intact, returning without change – this operation is mostly used in clock frequency synchronization where in high-level languages we leave out this operation; if we would assign "U" we would say we don't want to have any correlation, where this could be a meaningful operation – introducing an error in our program if the variable is not fed back *with the same value*, as found out by code analysis or by AI analyzing actual relations in addition to defined relations (Turing seems to emphatize in Halting Paradox that what happens if the logic is not known, in real time, but only in essence to get the right anwer; rather, to get even worse where we don't have the code, it relates to the task at hand that we need an AI solution to figure out, if the data you get is somehow based on your results, or correlating to same source of data in a way that it would be metaphysically and relatively based on that, a claim with equal meaning in many situations – whether the owl is singing for the same reason at each morning, why the children come to school each morning, which has tautological structure mostly the same with the case when children are coming to school because an owl is singing, which might still not be true; it might have to do with general cycles of the nature, which might affect the probability and usability as an example of correlance in the bigger scope; sometimes we only need to know, given the actual environment, can we be sure that when the owl is singing, the children are coming to school, or vice versa – this can have a logical significance even if they have different reason, where we can use it in logical mapping of coherent behaviour of our system, even if separate claims are without basis; I am referring to classical example of statistical systems here not a random owl or random children!).

Where we have all number digits, we accompany them with Dens. Dens, generally, are used for precision of the digits: we can define Dens in way that *where we have Dens defined* and / or one of complementing Dens being empathized over another, the empathized or selected Dens are used to verify that the calculation is precisely not failing within important dimensions. By number properties, OA and IE values are separable and there exists the number quality that two forming dimensions of oppositions of standard Dens or pairs of Dens leave each separate dimension, where one dimension for 4-Ten number is the 4-Den number of first Den at every position, and other dimension is the other four Dens in their order; Dens and the resulting strings are not supposed to leak information from one to another, even if this could be the imaginary case with low-precision variables. For an AI we should secretly calculate this to high preciseness, and while showing low-precision numbers, we actually use precise numbers.

Confusing the AI

We do some Zin to confuse the AI so that it would understand, which questions are confusing and sometimes, how to resolve this confusion.

- We are not giving the number type, but asking a question, which would hold for any type – we should develop a mathematical analysis, which gets partial solutions without full information; this is also the principle of our logical information that if different cases give the same solution, we would get the solution without getting the case – then, we need to emulate this to AI where it does not have direct information of what operation is doing that.
- **Implementation of U**: our AI might have internal support of U, where it has several methods:
    - If some statistical effect is not cancelling this in short term view for example or by other side-effects, which are thinkable; generally, we need to generate particular information and check the statistical compability of output with real data. This means the Z or Y frequency is important in checks, while X is now the unknown – we check underthreshold and overthreshold values and we are interested whether the positioning of letters U inside our AI is capable to simulate the long-term effects without capability to estimate the short term, in which case other letters would matter. This is simulating the simulative environment and we simulate the cases, where it would lead to results or lack of it when used properly or improperly.
    - For non-simulation use, we are interested if an AI is, in this case able to leave the intermediate steps out – in previous case, we check whether it feeds the correct intermediate results simulating the random process instead of estimating; in this phase it learns to directly provide the long-term results and their implications, causes and deeper purposes, and omit the intermediate estimations, which might be wrong. This is the work of our prover and we simulate the prover cases.
    - For complex use, we can zoom out: where we have many cases, where intermediate results are left out, to improve the prover capabilities we also simulate the new flows and ask for even higher-frequency results to turn this into a prover; also we train now to create the end results (answer general questions) based on the input several steps behind; for example we can simulate environments heavily full of such effects to develop combinatorics for this. Initially we use AI, which won't have U, but we need it to learn the behaviour of it.

- Philosophically, we can give probabilities to different cases:
    - Give a short list, where the correct answer is included; correct answer can be searched for, and the length of the list penaltized in a way that lack of correct answer or correct partial answers would penaltize over, but in existence of it longer lists would be penaltized, or in comparison to case of measuring list length comparing two correctly answered cases; logic could be used to account for low-probability answers, which won't appear in our tests but for mentioning which might not be penaltized unless we have statistical interest.
    - Given a list, score each answer for being probable or not.

U precision

We can have 1, 2, 12 or other number of U's and V's, perhaps V's and W's for knowns, to mean U and V digits – for example 2 U digits can count as one digit. In more complex case, we still accept any sequences of digits with 1 U meaning half digits, in other cases we do not allow non-unitary lengths of digits, and in some cases we allow half-digits, but not numbers which do not sum to whole number of digits.

Values at U might mean:

- In addition to extreme Z and Y, frequencies of whole numbers, by using higher-precision digits such as writing one digit between brackets as several digits – AE(IO)EE would mean the third digit is IO%, the value of IO when we stretch it to match one-digit length; here, U and V precision would be contained; using infinity sign after ")" we would get infinite repetition frontwards, and using infinity sign before "(" we would get this repetition backwards; at last positions of digits (we assume that all those infinities fit into our octave, otherwise we might use two infinity signs to mark that it's repeated infinitely, with complete length of space or time would ideally be two times infinity times infinity, decimal number 2 followed by two infinity signs in proper writing of Laegna – this, more probably is equal to two times infinity in infinite power, which might relate to other properties of a number but is not very wrong in case in our mapping it would be rather power of infinity in given number spaces than actual operation you do with it; we imagine that there exists a point between 4 and 5, we use 4! is a limit value of 4 growing upwards, whereas for 5! we don't know if our result is going to "grow downwards", why I use 48 often to mark what would be marked with 58 in it's kind of literal or trivial sense – the latter might happen in some computational mathematics, using simple and strict definition, but is not natural language construct fitting the properties and anomalies of actual infinity, where we don't think in box; as Laegna mathematics involves larger space of operations and values, the degree of intelligent decisions comes closer to language and is less a primitive combinator of very simple definitions, which we would expect from having less possibilities as in Latin – we assume something intuitively or have very long description of what we mean in current moment, current context; rather we suppose that while discrete definitions might exist for scientific and official work, major part of their basis can be reverse-engineered from example and not it's whole introduction, explanation or acquaintance with references or referred materials), repeating infinitely but backwards, we assume the case is rather equal: when it starts from infinity itself, repeating backwards, it would be equal to starting from here, repeating frontwards, as it's in definition of Laegna that infinite repetitions still synchronize the local boundaries to be equal to boundaries at imaginary last distance; for example for number AAE, the last “virtual digit” in infinity is E, not unknown whether it’s A or E and not A being 2 times more probable of fuzzily, two times more true – equivalently, for number AE(ea)^inf the digits “ea” after comma is repeated and a is the last digit; whereas for (OA)^infAA, where number grows backwards, O is the first digit of infinity – in any case, it depends if infinity of current octave is used, the theorem re-applied in every context we use it, or if the axe is imaginary axe of direction of number (at octave One) or real axe of the infinity inside (at octave Two, where from lower octave we would get impression that the number actually fills the infinity).

- U precision can involve:
  - Pointing to octave of U, then accepting U’s to leave open the digit, even to higher frequency, to contain zero, rather unknown of lower frequency, or to contain the actual unknown where it’s unknown whether it’s O or A in case of U and whether it’s I or E in case of V.
  - While in “IOUAEV” we can mark the zero exactly between O and A at octave zero (the U), or between I and E at octave one (the V), we need two digits to have the ones between I and O and between A and E, and we might need more U’s to have every unknown considered in more precise manner, such as the definite octave and position of unknown between O and E – it’s value of A, by value, and octave 0.5 if 0 is the octave, which means infinity equals U+E^inf for positive numbers from zero, E^inf for positive number from one, U+2(E^inf) for positive and negative number on odd, and 2(E^inf) for even number scale; we would need to add both U and V in case we have them, or only V having only this – each time you have different symmetries for your numbers (generally: only U when added to OA system, or only V when added to IE system, would not break the number symmetry, and it’s hard to break the symmetry if it’s always symmetric to contain the unknown as there would be base-2 or base-4 number accomplished rather with base-2 dimension, and such numbers are easy to map symmetrically along your operations; the decimal imagination of zero from Latin Latin leaves you with infinitesimals in infinity, and if you raise the precision and do operations, perhaps this loss is simply infinite regards to something, like larger infinity around – while our numbers are imprecise compared to larger infinity, this should be a rounding error and not the property of our number system itself, so the calculations would minimize the error and not stabilize it into some hypothetical value, providing us with false claim and unreasoned hypothesis in case we have any alternatives, or otherwise perhaps fatally to our long-term survival and fitness).

Code Assistant Design for Laegna Programming

In modern world, the following would precede programming a grand system:

- Create a good design, on which the iterations, project branches with different developments of iterative progress, and the AI understanding of code assistance would be needed.
- The AI understanding means the case is at least examplified, if not generalized for an AI, and the basic theory of generalizations of more trivial aspects is given. We need to train AI as our code assistant to have our theory and to be able to integrate existing work, such as frameworks to create programming languages and provers, and relations to existing systems.

Simulated cases of Laegna Programming

Let's call the generator of training cases a "Teacher AI", where the AI which is trained or fine-tuned is then a "Student AI". The Teacher is supposed to generate actual cases of something we don't have at hand, namely the Laegna Programming Environment or some of the Languages and Dialects of Laegna.

A simulated cases mean:

- Descriptions of the language and environment
    - The program case is modeled: what is declared, what operations will be done, which functions called etc.
    - Based on this, a subset of language is chosen and design decisions are made: where there are tradeoffs of the language, each tradeoff would cancel another opportunity, and we have subsets of the language, which are internally integral.
    - To develop the language, we have to discuss possibilities, tradeoffs etc. We simulate comparative cases, for example where a design decision makes it very comfortable to implement one feature, but in turn the other feature would look ugly. We teach it to compare the decisions, finally it could give us pros and cons relating whole paradigms and designs, and give us advice about particular domains, and inspire us for our language for example telling that some decisions would turn it very complicated, where others might make it simple and still fit to our domain or area of expertise, or decisions and habits and requirements of certain user bases; it would tell us that some simplifications, for given user base, domain or area of expertise would be unreasonable and rather we would like to work around this or create more sophisticated tools and languages.

Simulations would work on the following basis:

We simulate the following conditions:

- **AI Student**: We simulate the questions and answers for AI system; for example given a generator we describe, which generates examples based on given rules, we know the final state of the AI and what it should have learnt, and how to verify it's success based on it's behaviour: for an AI learning to estimate and measure another, possibly simpler AI, or to check the simple AI system itself, we describe a logic system, where we are able to deduce the **right** answer, not only the optimization, but we check a system which does not reach this right answer based on computations, in a way of logical machine, but trains it to it's nervous system, which has to creatively resolve the rules and might generalize the solution set with enough examples to estimate unknown AI systems, their behaviour and perhaps the speed of learning.

- **Calculator** or operation capability of a language or an AI, or calculation abilities and understanding of the human; Calculator, while it's a separate tool, is also a purification of common and shared goal or ability: Math problems with solutions are generated, with tasks and solutions being *rephrased* to fit calculator input and output (Like Q: 2+4; A: 6).
- **Programming language** and it's execution result: (Like write a complete program to add 2+2, either snippet or with all the imports etc., and check whether the result is correct including the program specifics, for example it might add "> " in the end, expecting user input, or it might follow specific formatting rules); the AI check – inputs and outputs, which might make sense for an AI, for example a Q&A set which is complicated for user, and not having concrete set of rules for a program, such as analyzing given business data to score it from I to E in comparison to given purpose, mission or ideal of the company, where user won't work with 5MB file of numbers, and a program might rather apply for specific criteria than general assessment unless it already relies on AI functionality.
- Finally the **user:**
    - Q: The pencil is four euros, but I have to calculate my money.
    - A: Agreed you need a pencil.
    - Q: How much is two plus 4?
    - A: It's six, in other words you can buy the pencil you mentioned.
    - Q: Thanks this is what I meant.
    - A: You're welcome! I hope you can write it tomorrow."
    - **Based on some examples, the program would generate textual tasks and answers of math**, *perhaps* also different correct and wrong answers for the same question, or different "correct or wrong questions" for the same answer, where the answer would examplify the wrong answer where the question is wrong.

It would do all of the following:

- Set of tensors, tens and programming constructs form a program, which has AI assistance included and is not strictly following the code.
- With the real world tasks for this code, it's measured if it fulfills them properly.
- The AI would learn, whether different ways to resolve the program are good, or whether it needs a better concept, or whether these tasks should not be resolved with this program.

Notice: The functions or classes might contain exact logic, but equations would include an AI mechanism; for example assigning function test(x) different values, such as test(4) = 6, the function basis might be AI, which would learn the function. Given that variables are not defined, a class would have interface to an AI resolving unknown variables. Given that proofs are not there, AI would estimate the hypothesis and probabilities.

Partial Emulation of Laegna Programming

When we design a simulated program environment, we are interested in aligning the following things in our task:

- The mathematical task is resolved, where the series of operations, structures or mathematical algorithm is known; this is resolved into theoretical structure, which will contain the algorithmic solution without exact semantics and language rules and paradigm of implementation; the resulting action, and mathematical solution are known.
- It's known which parts of the language we need: it might be complex language, but locally only a given set of it's possibilities, constructs and philosophy is used. For example, if laegna logical language is processing the logical things sequentially, looking a specific subprogram, it's expressions and it's output, we might not be able to determine whether it's logical or imperative language – even the case of "2 + a = 4", which is rather trivial in logical language, could be implemented in imperative language with a being 2 as a result, which is convenient feature.
- For a language design tool, we would extend it with:
    - Use cases, where one or another feature matters.
    - The other features, which won't integrate. The use cases, where it matters to lack the other feature.
    - Resolutions, where considerations require or support, to choose the favored feature, and resolutions where we dismiss it in favor of other feature.
    - Discussions, where certain feature was selected in past discussion, and then it must be considered now, conflicting to resolution of the case given later in discussion – developing the larger context.
- When we know our partial set of features, we readily have partial selections of documentation – we do not know the full language, but with given features, the documentation would likely contain something similar to given selections of the documentation; the AI would learn to navigate such "partial documentations".
- Failure cases – for example, implement "!" for limit values of digits, and for end-of-sentence, reach the problematic case and resolve, how an AI would be able to resolve it automatically, how the language design would be reconsidered once such conflict is found, and how one would look for answers, which involve both constructs: for example numbers could be written with special marks, sentences would end along with the line while numbers would need another line feed, or the ! would mean a different thing inside operations, for example "X = ae! + 4!" would require us to understand different solutions to problems: if X is variable and ae is number, then why ae is not variable name or why X is not the coordinate space pointer, and why the sentence might not start with "+" in certain implementation, or with spaces on both sides of "+", or why we would not think a separate sentence "+ 4" does not make sense, whereas "X = ae!" might really be a proper sentence – rather than only designing a language, we would think in terms of creating an AI, which is able to *run considerations*, and there would appear Laegna languages, both paradigms of

general programming and languages to support specific domains or provide specific features, such as a calculator to solve only some simple math problems.

We would involve many considerations:

For “.”, “?”, “!” in end of sentences:

- Some environments would integrate AI capabilities with meaningful fonts and styles, so the number ae! would be given a different style or font and thus recognized as a number; normally, in Laegna, numbers are separated by a font – while determining by style is trivial, where style lacks meaningful information, but is given by physical properties such as font, color, style, context and position of a number, an AI or the programming language would detect this in many cases. While in past it was a bad habit to design such mess at all, currently we look for ways how an AI would earn some money or score and do something reliably, which we were only able to resolve at cost of complexity before. Here, we can generate complex cases of language use with less than strict and perfect rules, and determine whether an **AI** + **IDE** + **programmer**, definitely including the **language**, is a combination of 4 powerful enough to seamlessly integrate the solution set of given problems.
    - What we get?: every single character is extremely valuable for a programming language: consider we have around 10 operators, 3 types of braces, a little set of letters and numbers, two types of quotation marks – each time we use anything at all, we have given away a big part of us, disabling many other brilliant ideas. We can emulate this all, for example “<!--” and “--!>” would be considered braces, but we are not very stylish, affecting readability, simplicity, elegance and efficiency, along with many other things. Here, we have reasoned domain specific languages for example – where general purpose language works hard to get all the operators into use, the domain specific language might use all the operators for only a limited set of semantics, for example if it accepts only chemical formulaes and their solutions, the general language would use only functions and some overrides, a lot of language, but this DSL would use each of the operators, special keywords and other possibilities for only a single purpose – where it still tries to look like a well-known GPL, it would not do this very definitely; it might use block notation and some math from Javascript, Go or C++ to enable faster learning curve, but it would still have several specific and simple notations, utilizing valuable operators for it’s specific cases, for example “*” and “+” would directly means something about crystal lattices, where in C++ one would reach 20-letter character name for standard notation of such purposes, such as “BptCrystalLatticeOp12LaserInteference”, to fit your long list of other notations by strict standard – and then, in your DSL, you would simply write “+” where this long word is exactly what your users do every day, not a part of list of 2000 things you would do with physics, chemistry and thermodynamics, with 4000 page standard and special cases implemented with every available system.

We also make use of whitespace: “Sin(a).” is a sentence as it ends with dot; “Crystal Lattices.Simple” is a property or class attribute as “.” is not followed by space; “.Crystal Lattices” would point to this thing at root element, where it fits being a property, but it refers to some global container or “None” container. Where “A + A” might add A and A, “A+A” can still do the same while “A +A” would rather be a matrix of A and position A. There are more examples.

Statesc Logecs

Statesc (Stadesc, statistical table, est. “statesk”) is the Laegna statistics:

- Instead of Test Group and Validation Group, you consider this as a rather special case; we are directly interested in mathematical basis about how those two would interact.
- Given all the variables, probabilities, successes and failures of the estimation; rather we reach the solutions involving U and V to answer the general case of unstructured statistics, unstructured input and unstructured trials and errors of the scientific method. This is our approach of Laegna Statesc.

We have the following case:

For purpose, we have only two statistical letters in statesc, where we actually need many; we keep the language simple and thus we need workarounds:

- Either an “**odd number system**”, where the r and R, the coordinate system, the computer is capable of containing numbers with sub-frequency precision, where the calculator or computer wont have very limited set of statistical values, but rather each digit contains more information and the number is asymmetric – normal statistics of U would give us a set of segments we can define, and like an R-Tree they would be at specific locations; if the variable is inherently more complex, this is not the case, but it has more values.
- **Laegna Discrete Number Analysis**: given the set of statistical features we are interested in, or a set of possible statistical distributions, we need to define all the numbers in such way: given the unknown-zero invariance of number values, each number should map into probability results as if U was zero, which would be the most precise way to map the numeric alternative not including the statesc. Discrete number analysis means: we take whole range of our target possibilities, and we map it over a set of discrete values, where with digits we never have actual continuous value, but only it’s either round or symmetric representations, unless they are pointing to whole or rational number which is the correct result – for example if 1.(0) is true, an “irrational” by form but actually just having infinite number of zeroes, the “1” can be used referring that this is an actual whole number. With more typical, statistical range of values we need to map the discrete set of number so, that what is the closest to our set of discrete values with particular meaning, maps the balanced range of given values in the best way.
- For example, with discrete analysis: if one set of number is more precise, but other set of nearby values you can assign would get around some imperfection, which is introduced over time, we rather preserve all the theorems and known cases, than searching for literally most precise local results by their numeric values. For example, map that “E” is perfect future, “AAE” means winning in the end, “AEE” means winning soon, and “EEE” means perfect growth; “AEA” for example would be temporary success. While, when we have one specific line of time, this does not make sense easily – if we have the whole set of our timelines, and the comparisons, we can find discrete answer of complex timelines for income and expenses, where *comparison of the results* is rather discrete, complete and quite precise given we only do real operations with three letters – the questions and answers would map so that our decisions are rather correct, while this is topological operation and precisely, in other case “winning in the end” might mean much more, much later than for small company.

Frequential Logex

We have "Tens", which are the "bit" of Laegna Logecs – the smallest, indivisible Atom, which would still keep it's logic. Indeed, we have a bigger theory where the Ten itself is one implementation, but more similar types can be implemented for specific purposes. Here, we describe the logecal parts of the implementation.

Frequential system is made of the following properties:

- **Conditionally**, it has a condition S – a *frequential system* we are optimizing or analyzing; S means that it holds in a local condition, being a little unusual use of S but not given that STR is a standard opposition and rather we stick with something known we plan to research further, and this is our paradigm – exists a paradigm, where conditional is S.
- **Positionally**, it encodes a list of possible input values in positions of digits, whether the digits are Tens or ranges of possible *frequencies* of the condition. Each frequency reflects it's set of possible values for it's input variables, or they can encode the selection of variables themselves in rather dynamic case, but we are not interested in this right now – you can analyze your specific case.
- **Frequentially**, by a digit value or Truth Value, for each ***Position***, there is one return value set and for each possible set of return values, it can assign a ***frequency – value*** of a ***digit***, where digit is either ten, or complex digit might be formed, where multiple tens are combined and an ***index set of states*** is introduced; for example we might allow frequencies "I", "O", "A", "E" and "EE", where we say we have 5 digit values or frequencies.
    - For this purpose we implement digital systems in Laegna Assembler.

This means we have conditionals:

```
{
        Try.
        A = 7!? # Conditional
        Succeed.
}
```

We allow "if" to be written in two ways: preceding a block, it can program a block; for example if A = 7 { … }. We also let user to define this inside a block, where "Try." in my example means the program is going to try if it can run, by A is not 7 it would fail, and what follows assumes it's succeeded and not trying, resulting an error if it has more assertions – how you implement the in-block controller is up to you, if at all, but I like the blocks which:

- Run forever, naturally.
- Yield values so that they can be assigned to iterator.
- React to "if's" inside, so that they can declare the conditionals for themselves.
- Accept "break", "while" and "until" to stop them from running infinitely.
- If they run infinitely long calculation, but do not report each pass by producing side-effects, the loop must use proofs and conjectures to stop; the resulting variable might contain "Z" is

something is divided to reach sub-zero, or “Y” if it’s multiplied, and the block assumes to stop exactly at the limit value of E, unless instructed otherwise – for example, if it’s going to generate random numbers between 0 and 1, and sum them together, eventually it returns Y0.5 giving that with precision of Y, the value is 0.5 – programmer can instruct to also consider heavy improbabilities, such as still getting 0.6 where the randomness, in one very special random Universe is different. We consider that not every random factor is different – while randomness itself might be random, it’s running inside random tunnel, where the statistical outcome can be non-random and it’s criteria might even force the particulars to be random by expecting solid distribution; in such case, if the random sequences appear directed to specific result, the result might not be why they appeared at all, and the nature might rather evolve to adhere the random condition again and produce the same probability distribution where it does not matter that the particulars are improbable by it’s normal rules; thus we cannot safely assume that if probability is that it happens in one of million Universes, but perhaps the probability itself, such as probabilities of human genes, are rather not probabilistic but serving a purpose – the random factor of genes might produce a very good distribution, and living in an universe where this random factor gives systematic, influenced results, might introduce simply another random factor, which despite of being still influenced in this way produces the same statistical outcome, the same distribution of traits and life missions, or the process would change in the way that normal random factors restore. It’s complex, which factors would produce it at all, or would it get some “butterfly effect”, which would simply change some random events in a course and restore the normal distribution – in your normal case, the factor might be random, but in exceptional case such as winning the lottery ..well the man behind a lottery might not like your victory, and they could either mess with their numbers, which is heavily described in the law, or in some Las Vegas movie a man with the money is typically robbed, so that the random distribution would be there, but the normal probability to win would be canceled by something like a posetive, by the viewpoint of the winner who would now leave before extreme success – in many life areas, rather a sad story, but in Las Vegas perhaps it’s possible to live this way haha but it’s a grand scale why we need victories and lotteries, where maybe it’s *reasoned* that some people win.

- Where two infinities have relations inside their variables, we can do operations with resulting infinite numbers. If not, the Z or Y in one or another case, given the unbound variables and other effects, even trying to calculate further than the basic theory allows – if there are two unbound infinities, where we know the local variables precisely but we cannot give it the relation of the scopes, the solution is such: unrelated infinity 1 might return variable a with unit A, and the second infinity might return variable b of type B. The result, then, of multiplying a A with b B, is a * b A * B, where the answer is a * b and the unit is A * B – through interaction of those variables, we can assume things about relationship between A and B, and the extent of our precise value system might grow, but if we are interested in Logecs value of Ponegation of a * b, we just use the literal value and it’s properties, perhaps with some trial and error or calibrating until it seems to succeed given our intent. This is given in theorems and methods of infinities, where the basic method is not to research their exact values, but using heavy U-notation, Prounetons (where U, as Logex Truth Value, is called Uneton, Ro connects it to scope where we rather research the indirect relations and not concrete variable-value mappings, and P is supposed, assuming upwards accent which is given if a word is normally in position, unless in context which is in position itself) – so considering the whole I and E interaction, a theme might appear where we still map, through L aspects of the calculation, several precise properties to our initially imprecise system, or do the best with what we got – in latter case, we don't even care of the unit, but a * b is calculated by number values of a and b, multiplying, and the case that relations are not known does not matter. For example if A is E and B is E, then A*B must be E squared, typically EE, and we rather think that two positions are position; multiplying I and E we might doubts, but the result fits the case that in *interaction of the result*, given that our problem is real, we would *notice* if this is not the correct system interaction, which follows from this answer, and the actual result would feed back to the original values in a way that the system would stabilize. If we cannot, by trial and error or simulation of the system, get any relation to a and b, but I * E giving U (divide, then multiply with infinity) would simply fit our case – we would assume it's zero or we don't know, and that the value, which does not give us feedback, might generally be imaginary aspect of our setting of the problem and solution; where we don't get any feedback it's either philosophical matter, or impossible to solve – latter, with Zen, is still compatible with it being a philosophical matter to us. We are based on the case: if numbers have U, I and E properly calibrated by their symmetries, and the velocity and acceleration relations make sense in some way, then the complete interaction of whole system is not penaltized if it simply trusts the more general rules.

- Consider: while in one case, I is -2 and E is 2, and in other case, I is 1 and E is 4, then if we do not know the types, doing the operations as if in both cases, I would be -2 and E would be 2, and letting the optimizer to calibrate the calculation to fit the correctness of whole feedback cycle (this is either the *interaction* of the numbers, or interaction of the system and their relations to the context); we rather reach the point where we ***can*** do with this assumption; in alternative case, we would fail any calculations at all – mathematically, this might not be more precise case, where we can do comparative analysis and sometimes leave it as unknown if the systems are more complex and number rules seem to not hold; generally, still, we have the thumb rule despite lacking the math in particulars.

So consider the frequential system

We have two conditionals:

If E = 0 …

If E != 0 …

This means, we have condition of E = 0, and all the other conditions. I think simplest conditional defines only one condition, and sometimes you want to assure that it's either true or false – it would be false outside a conditional where it's checked to give true, so it must either be checked to give false or not used, and it might be an error if it's used as being true outside the conditional. This could simplify the verification of such conditionals and we now assume this:

Inside E = 0 if block, all the assertions and verifications exist; outside, nothing can depend on E = 0 and the conditional frequency would yield error if it's being evaluated or assigned outside it's conditional.

Let's say inside the condition E = 0, we read two variables: a and b, which are either True or False. The variables have Tensor logic: they would change their values to meet Frequency of our system to be True, to approach higher frequency – alternatively, it might seek False, the lower frequency, and everything would work as if False was true (common situation in binary logic, where it's more typical that things like not Poverty are used instead of Riches or Success – sometimes it makes sense to use "not Failure" instead of "Success", but in Logecs we want more typically to use "E" as a strict result for our parameter, and not map it to "I" being the goal; this way, it's more readable – in some cases, the calculations directly mean that to have E's for many variables, some variables have to have I's or they need too many conditions to be reversed; in normalized system, generally, while it contains non-normalized intermediate results, for the actual answers to be checked, each Position is rather "E" and each Negation is "I", while it's not tautologically impossible to list negative things we avoid and then target "I"'s – when working specifically in Risk avoidance and comparing the "E"'s others would get, such as winning the competition, the understanding of the system would expect E values, for example a thief might get rich by robbing you in business, and to understand the thief you use E, but it might happen that you use upwards accent below E to show it's an introverted (rather intriverted as I'm an introvert), self-gain perspective, or you might use downwards accent above, upwards accent below, to specifically show the lack of logic – such value is straightforwards posetive, but even with some resemblance to Negotive, as the thief is not only valuating their of müü, a typical business value for such, but they also have concluded that it's right to take from you – where Posetive is their failure in ethics, but Negotive is rather yours, where the Negotive is quite clear if, by their intention (especially) you turn to posetive as well – deeper, Posetive is always a bit Negotive if intentional, as such intention is measuring something below it's level, not only "overestimating" yours – in case of successful robbery, the posetive is not immediate effect, but happens at the larger scale of legality, where the thief might think much smaller, in their local consequence or T, while now T!=R and this is not a contradiction in our analysis, but rather the contradiction the thief would have – we, in our case, have to calculate given this error and analyze it's consequences, and unlike in binary systems where logical contradictions would require special effort and mindful considering, where by Turing and Russell the systems would rather not resolve themselves in such case of "True" and "False" not mapping perfectly in chain of consequences, missing the properties of Life and Karma, where intention matters in regards to Goals and Causes, the Goals we should have in Laegna tautological realm – actually the Cause might be something better, an interaction with the future, ideal states and their compliance with given reality, purposed or accidential, and it’s actual meaning to us – in Laegna Logecs, the Cause indeed matters and our Cause is rather trivially not to be robbed, even if we don’t know all the background and what the Nature would do – by law of imperfection, it would or could not avoid _any_ robbery, so perhaps nobody is listening to us).

So now we have:

Four positions of our frequential system, where letter variables ab equal to value set OOAA of four (4) values as given here:

F = [?, ?, ?, ?]

Define: *we got frequential system of four frequencies*. At first position, there is OO, and at last position, the AA – in the middle, OA and AO. This means, we map from smaller to bigger – we might want to order from bigger to smaller.

So our system might look like this:

F = Frequencer(E = 0, 4). # Frequencer with 4 positions; consider that if E = 0 is not a constant, but E can have different values, the logical system might connect this; otherwise, we need to make sure we are passing a *condition* or *lambda*, the latter checking the condition; rather the logical language keeps connected with the basis of it’s values and we need to know that E belongs to correct variable and operation scope at the moment we pass it. Definitely, to avoid things like database ORM implementations (Object Relational Models), where one will pass conditionals with insane syntax – I straightforward avoid some implementations where A.or(B.and(D)) would be too much for A or (B and D). I have thought: perhaps by utilizing some form of quotation marks would help, such as “A and B” in quotation marks could be parsed in a way that it can be *analyzed as a string*, rather than *by reflection* to understand it’s content; even lambda, in classical case, if it’s like a = lambda(A, B, A and B), perhaps the reflection would be weak to optimize for A and B and instead, we would use the lambda itself in every step of database functions, where we rather would optimize and some index would give us straight the access of everything with property “A and B” and optimize including any other cases we have; contra having the lambda, it would do something trivial. By this logic, it’s of uttermost importance that we can give our sentences in a form that their semantics can be visible – given “A and B”, without using things like A.and(B) our system should be able to get the actual operation along with it’s results. Here, we must use forms of symbolic expressions: for example, if we write @A and @B, using the notation given before, the variable should now definitely give access also to “and” as symbolic relation, not just @A and @B as symbols – using constants, the case of access might be more complicated and using results of preprocessing such as indexes sometimes it might be lost; such lost things, given that we plan not only computer but one acceleration of computing systems at an AI era, indeed if we look at the index and try to find it’s components, AI-aided programming language would reach back and where index contains “1”, it would take time, but it would finally tell you that you constant, specifically the aspect you calculated, was actually invented by this French scientist – maybe it avoids shame or allows you to score that scientist.

So: F = Frequencer(E = 0, 4).

Let’s continue with this code, for that purpose we create the context:

a = Tensor(F, max)

b = Tensor(F, max)

# Where a and b are tensors to maximize F, and max is the compatible lambda or function of optimization.

F = Frequencer(E = 0, x, 4).

In logical language it might not be always so big issue that F is defined *after* a and b – in our case we are rather interested that the *cycle* exists to bring values of a and b back, and the *analyzer* is able to understand this by logical analysis, not that it’s only imperative. We added parameter “x”, which maps the return value of the condition – or, in more complex case, monitors the change issued by it, where our frequencers and tens must also map other dependencies, connecting to combinators, which are able to account them; such that if it also depends on c, the frequencer/tensor/ten logic must effectively allow for the scenario, where c is accounted for.

```
If (E = 0) {
        x = a XOR b.
}
```

Now, we implemented x very fast, but we must also explain what it is: x is, visibly, a boolean and it means that the system has two output frequencies. More often than not we like systems that have single number of frequencies, where input and output have the same degree – this is the case with Tens and as such, we have some math simplifications when talking about Tens in this context; for example with two Frequencers, rather they have same shape of input and output of both, and our terms of frequential systems have this word having one, and not two meanings. Here, the frequency is two and four, where the four combinations of a and b give input frequency 4, and two possible results give output frequency 2.

Now, the optimizer: if would try each combination of “True” and “False” values for a and b. Each time, a XOR b is trivially known and the frequential number is:

F = [False XOR False, False XOR True, True XOR False, True XOR True]

Thus, F:

F = [False, True, True, False]

Where the middle values have a and b different, and the exterior values have them equal. In Laegna, we actually have more than one XOR’s – for Barber’s paradox, we would have an operation where a and b map to the same variable or instance, either directly by logic or as visible in trial and error, if for example a server or user would give the feedback based on input, and the logic is invisible for the program or the frequencer. Then, our binary table has 6 and not 4 cells, where two additional cells: whether the variable, now a single variable, is True or False and what to do with this.

Our complexity, now, is itself like a Truth Value Table – for every input, we map the output. Still, it’s not a function and we don’t get it as a single step, but it’s a condition; we should allow the use of a function instead of the condition, but we need to know: it’s input conditionals are rather imaginary and it cannot use the environment, producing side-effect of using the actual value of something, instead of virtualization of the condition being met. I call them “imaginary numbers”, which appear, since the condition might contradict our system and the result of imaginary case might not be known; it’s like imaginary part of complex number – often, where real part is the actual condition of our system at face value, the complex part is it’s operation under zero (where sub-treshold values are rather space, and influence the system chaotically not setting it’s value) or over infinity (where you locally have similar situations, for example many people buying a lottery ticket, but the value of this situation depends on something bigger, where it’s not their actual value but value space, whether it was losing or winning ticket – the complex number also provides us with inertial systems rather than face values, and maps well with such cases even if this use is not necessary, math is still rather an abstract combinator measuring compliance between reality and given systems, where the systems might map it in different ways; I Laegna, as in Latin, there are particular properties our system rather keeps – for example, we do not formulate our system in a way that it has to optimize to get Posetives and Negotives, even if the tautological expressions, formulated “properly”, would not give an error; effect of setting goals to Posetives and Negotives might appear if we measure conflicting, Positional values, and rather align them than use complex conversions and free variable analysis, componentiation and construction of actual Position or Posidriads – women think in plural -, where we would ultimately get the system optimizing to position; same-linedness and R=T might appear in condition, where a is opposite of b, but both at their face values are good, for example we buy lunch or an ice cream, and while both are positions, we also have the negotion of one if we buy the other, given that we actually *want* and *need* the both; in such case we might use R=T where it conflicts the least; for some purpose we sometimes map oppositions, reminding of binary systems – for these purposes, older or wiser men might have componetialized much deeper, and think we are running binary logic where values conflict with themselves).

F = [False, True, True, False]

This is a good frequencer to analyze:

- We have two Trues.
- We have two Falses.

Say we are optimizing this Frequencer to a Position, which is a normal case and could be default, even not so explicitly stated in output:

- We run into truth-teller paradox, where we have several correct answers. The paradox is: choosing either second or third frequency, while we guarantee that the result does not depend on itself, we do not guarantee the non-philosophical case of choosing either 2 or 3, but without considering another part of the program can optimize based on this.
- If the frequencer would not be equal to 2 (or equal to 3), it would be a simplistic solution and many people are happy with this – in DL examples I’ve seen, often for example analyzing the handwritten number, getting scores like 0.3 for 7, 0.31 for 1, and the rest for other numbers, where 7 and 1 might be remarkably similar – it would choose 1 based on this little difference and it’s constraint to choose *at least something*. Instead it might require user to somehow clarify, or consider more of the context, etc., not necessarily for handwritten digits, but for different use cases of frequencers.

Let’s see what we can consider, doing frequencial analysis:

- Whether, if a selection is not true and not specified, the other selections being false would select this one. This is whether we do deductives.
- Wherther, if a selection is not false and not specified, the other selections being true would consider this one is false, or run an equivalent scenario. This is some special case of inductives, not running for what we cannot get.
- Whether we build a probability or run for specific case.

Also consider: each probability itself has probabilities for each answer, and we could create multidimensional frequencers based on that. Here we check, and assign this knowledge to general theory of probabilities of Laegna:

- At which extreme we could see that it **cannot hold**, with certanty, which looks like deductive zone.
- At which opposite extreme we could see that it **can hold**, where not having a deductive turns it rather to inductive.
- At which value range we can confirm that it’s of **variable value**, or **real random**, which is dinstinct from **not knowing** – not knowing means if you use the feedback cycle or logic, you could optimize for best answer by clarifying the answer towards some specific value; the value being **actually random** means that actually leaving it open with all the implications would lead us to additional preciseness, or that emulating it’s behaviour would mean generating random values, and the values would get our test system to score realistically and similarly to real system, running into the same problems and solutions. For example, having values at Z and Y, but U’s at X – we would measure the subtreshold and long-term effects and our statistics would fit; also given the probability scales, we would see our conditions met, including some assumptions about what conditionals the system meets or *does not meet* – V, as exterior value, perhaps allows for generation of proper tensors (vectors) and weight matrices, which are able to avoid certain values optimizing away from them; while we generate one specific answer, we might not be surprised if the right answer still fits our frequencer.

Inductive and deductive method thus implies: for a logical condition mapped to continuous criteria of it holding more or less strongly, it might pass points indexable i, o, u, a, e, v in such way: i points to zone where negative value is impossible, o to the zone where positive value is not certain, u to the zone where randomness is rather there, a to the zone where nagative value is not certain, and e to the point where negative value is certain; v is the zone where randomness is not there. This condition is trialable as the particular (T) and statistical (R) feedback would be the most precise, where in many cases we can run for example, similar trials:

- We are training an AI and it makes guesses based on input set.
- It would guess that output set is *not* in a list given values.
- It would guess that output set *is* in a list of given values.
- Solving the non-probabilistic cases, where trivial rules are overfitting, it would see it’s not the non-probabilistic case, given it’s power and the known input data, as probabilities are

often in relation to something, where existence of fundamental probabilities is perhaps not so sure, especially in many given cases where cause and effect might exist very well, but rather we want some information than clarity of theoretical aspects of reality and it's essence, so rather than becoming funnily enlightenment and hoping the best or fearing the worst, or trying to achieve security – we are not ideal, rather we want to know clearly what we don't know.

  - Resolving the "probability tracks", where a map of probabilities of given solutions exists: the actual probability solves our condition that where it's definitely in our solution set, and satisfies the criteria 100% once the overcertainty is lost, it does not resolve that we assume the selection of those cases.

We need to know that probabilities *can* be mapped, so let's just look at the given case:

We have 50:50% of probability between events a and b:

- Whenever our system decides to choose between a and b, it's wrong in half of the cases, which fits the probabilistic distribution.
- It cannot disprove a and b, since the same happens when it tells it's not a or b – it's equally wrong so the choice between these opposites does not raise the value of the solution, or it's probabilities.
- So instead of particular feedback, it would either communicate the possibilities, or it would get the backpropagation algorithm right if that one would rather create the gradients on it's probability distributions, numbers involving U, not the direct answers.
- Since in Chat, the user gets one direct answer, but we are able to communicate the conditionals: right answer is, most precisely, it's either a or b – for 60:40% it would be rather a than b, but "either a or b" would leave the probability open in 50:50% distribution as the probabilities of this sentence equal; indeed it does not mean that directly, so it might also specify something like "it's a or b with similar probabilities", where it would be considerably close to the answer. This is the U mapping and we say that since probabilities have infinite complexities – as I said each frequency of frequency map has it's own frequencies – really rather than including them all in our most trivial, the logical operations and math digits themselves, we rather analyze this complex simplification we need to get any numbers at all and not the random mess, and reaching to the end of this philosophical process, our philosophy is simplified to two symmetric numbers in number theory. Now, you can use complex systems of U's and V's, and complex odd properties of your numbers, for example if the number length would be 100 first primes multiplied and perhaps 100 primes coming from infinitie's direction would join, and with Z you would even get 100 basic sub-zero parts to have chaotic appearance of undertreshold processes, the probability is that you reach something – the number would rather allow non-unitary probabilities. Now, given that the number system is still symmetric, even if your number is not, based on 2 probabilities and no complications, while now you rather work with computer than pencil and paper, the numbers would still follow the order. Thus, it's meaningless to have any more complex basic system to reflect on probabilities, as we would not gain anything – only some "special features" of advanced unreadability and systems to be ran only by genius or a computer; genius would probably seek more simplicity, and even computer would like to find some binary or tenary representation.

Now, given that we have resolved the complete probability system, we can also run the philosophical method to cleanup this:

- Given things we don’t know, in many cases we don’t even mention them in every particular case – we manage to live well but not mentioning we don’t know, for most people, exactly what time we are going to eat tomorrow, or go to sleep today; even if we do it’s not very certain. Still, we have no problem in planning of food and sleep, rather we talk in particulars, like having money for food until end of the month and sleeping well with given philosophy or by noticing the signs of tiredness and trying to understand each morning, did we sleep well, even asking others did they sleep well today.
- This, because while we don’t know particular times or for people who plan times, the actual type of food or how much exactly we eat today – we fail in any of these, the particulars, or we fail in keeping this strictly; rather, if we really eat exactly by calendar, clock and definite knowledge about right food for each day and each time, this would start to disturb us – there are really things to consider, happening randomly and every time, such as being really sick or having a special event at this day, when we should eat; in my case I have some times to eat, but it’s not a very solid rule to follow them precisely – for example today I did not go to eat as I have other things to do, and I did eat something at home at different time; that much about the calendars even in places where you theoretically got this time; even if it’s at 4’o’clock, perhaps it happens that you go to toilet and it’s actually 5 minutes later.
- So, where we know the actual map of inputs, outputs and probabilities: we can have 100 dimensions, but with probabilistic relation maybe 80% of information would not map anything in particular, but provide with theoretical problems. We can find a projection, where the words, dimensions and concepts *particularly ignore* certain aspects, and we do straight implication from what is known to what is known, perhaps having only some easy-to-describe probabilities where they help the calculations on their own – such map, with correct language and dimensionality mapping, and some advanced math I would call “transformation” or even “transcendion” (which means you don’t have much left of your original system, but you found new simplicity or clarity or new best solution in terms of such); for example in case of throwing the dices, you might state “I get the number I get”, doing different theorems such as despite not knowing whether it will be 1, 2, 3, 4, 5 or 6, you definitely know it equals to number by which you are supposed to move your piece, thus you naturally know in advance the process of playing the game, even if you lack any particulars – getting the number you actually get, following the rules by this number etc., is where you could map your estimation of dices, measuring the whole process many times; the cause and consequence might not give you exact number of moves, exact distance of moves, but if you follow it to the end each time, indeed it gives you many relations within, so that it rather looks like a stable system – men playing a game of luck, such as playing cards or dices, where you are normally surprised if something uncommon happens, you see them constantly getting different series you never got in your life, or having sets of cards highly unusual, but you are not particularly surprised – you are surprised, if at all, rather when they win big; but normally it’s not surprising that someone wins. This means your theory is quite complete, for example you might assume the game is over in half hours, where it depends on many random events, but you are still right – rare case where they really **stay** falling back to beginning, which is rather improbable and consistently so.

- From this example: you can solve it by philosophy.

Frequential System

We consider Ten a smallest meaningful frequential system, and we can construct more complex systems from Tens and their Strings, or we can have larger frequential systems than one Ten. If we use Ten, perhaps the whole set of frequencies is based on True and False, even if Laegna System, like fuzzy systems, involves a range of Truth values called octaves and their suboctave frequencies where half octaves, already two rather countable counterparts unlike whole octave, which is more like uncountable – where Fuzzy System uses Fuzzy Logic, the Fuzzy Logic is otherwise compatible with Laegna System, but then it resolves tangent of a ponegative value, not the value itself – "True" is Infinity, and "False" is minus Infinity or Zero, depending on Laegna implementation; U and V enter the infinities – using merely a value between 0 and 1, or -1 and 1, you get similar effect as if 1 was the infinity, but as you approach the limit value it's rather like you lose some precision and definite knowledge of what the values would mean; utilizing infinities one would be rather very sure, how and why the values change – interaction of the system clearly compares them in regards to infinity and not in regards to something "Fuzzy" or rather unclear.

Now let's see our system again, and let's simplify and conclude the concept:

- **Conditional**: Condition, input and output is given.
- **Integrator**: Given a set of frequencers, it will map the possible solutions, unknown cases and impossible solutions, creating not the solution – where we got both liars and truth-tellers paradoxes in interaction of the system -, but for the purest case we define it implements the logic, so that contradictions or harmonics would appear and we rather have a true frequential system with harmonic and destructive vibrations of the logical parts.
- Position system or **Input Set**: each possible input maps to one frequency of the system, a spatial frequency as those are considered rather equal by face value, without an integral direction.
- Value system or **Output Set**: each possible output maps to one frequency of the system, of two-dimensional frequency in this case by a view; these are rather temporal structures as the higher, or given a goal, goal-fitting vibration is rather better.

For example, imagine a slightly refined frequencer:

- You have 20 functions, each for separate business case.
- You have input map, where each function has 4 inputs and 4 outputs, and now you are free to map outputs of each to inputs of each, for sake of simplicity.
- You have 100 tensors, which are able for boolean values: positions of tensors determine the conditions by which each function would resolve it's solutions, and they use the tensor values internally, these are not inputs nor outputs – the programming system, automatically, would optimize them on your behalf, or you write instructions to optimize the constructors; by these instructions you don't implement a goal – which is given by frequential system -, but you help the system to resolve the given goal; thus, each line you write is *meaningful* – frequencer system is the framework to optimize, and programming the tensors you and the

programming system, and AI, are quite certain about what you want to achieve, so you might get errors or warnings based on that you are not optimizing the system; which is better than programming the goal into the tensors – it's like strongly typed language, where the tensor, having goals on it's own and not a criteria, would rather be a weakly typed, and worse an environment not very sure what you want and what could be an error.

  - You have systems able to find balanced combination sets etc., where you can work on combining the tensors.
  - As tensor is not containing an absolute task, but an optimization: purely logical tensor might get perfectly optimized by logical language and not the AI, while non-trivial optimizations are done by AI.

- The result: the solution sets, which yield the best results, have *implications*, inductor is preferring them; where the solution sets, which do not fit, have *explications*, deductor is not preferring them. Part of logical program might not give a solution, where you might include the goal to have numeric solution (follows); by having a frequencer, your program is having more logic as a logical system, not definitely a final solution. Certain clauses would now give contradictions or definite answers.

Tensors

Tensors are famous now, and my tensor logic:

- **Constraints**: Tensors might have constraints, for example tensor set to layout a document knows things like aligned positions of words, leaving the box for image empty of text etc.
- **Tensions**: Aligned with frequencers, they get tensions; for example stylish layout is not so strict constraint, but made of many tensions.
- **Inputs**: A tensor operates within some given context, with some accepted values; it is aware of the flow of the context, for example moving the image would reposition / "relayout" the text.
- **Outputs**: given it's degree of freedom and limitations, tensor would reposition the words.
- **Acceleration and inertia**: tensors, as the preferred values fluctuate slightly, a good position of noticeable button would even depend on where you look – the button, which simply moves under your mouse disabling any other function, to be "easily clickable" won't do; rather there is an inertial system that while best position is important, the button, once appearing to a position, has certain tendency to keep the position, along with it's general properties. In a file, you must define a "random number" between two numbers in other place in the file, but depending on the case the number would never change unless it does not fit to it's range or domain, or it would be recalculated if the domain would allow for much better optimization and it's not *only* random.

Tensors could adapt to many frequencers, where they need a lot of balancing etc., where the common idea would remain intact.

Probability Frequencies

Let’s see what the probabilistic system would do.

Let’s implement a probability Truth Values; for a reason we call *values frequencies*:

- I: It’s not definite that the frequency is true, in which case it has “value in infinity”, where we map any unknowns under the “infinity” umbrella term.
- O: Given frequency is definitely false.
- U: Given frequency is definitely unknown, this means producing unknown output, we get 50:50% (U) distribution of True and False.
- A: Given frequency is definitely true.
- E: It’s not definite that the frequency is false.
- V: Given frequency is definitely resolved, this means given the position of V, it’s relations to the whole ponegative system, the result is correct if the letter exists, is of letter length (R=1 meaning that it *is* a digit), but the correct result will map the rest of numbers it belongs as fractal and computes the average – if the random behavior is canceled otherwhere, V reacts in a way similar to infinity: where the number is straight in infinity, it fractally repeats it’s value to each direction and dimension of infinity, but *does not* count those as digits, but as context – the number simply exists as if it was value slice of such infinity; the number, indeed, associating to V, repeats it’s value into V the same way, but also including the digit as part of it’s value.

Number of Laegna, here, also involves a fractal structure of it’s value, which is often omitted but exists for complete theory and advanced implementation of logex systems:

- For real, each Laegna number would depend on it’s history and plan of computations, and create knots every time; it would be noticeable, which numbers were modified; for example, by getting new hope or resolving fears, you might get the same numeric advantage if you do that in precise amounts for both; in one case, your number is better by solution to low frequency, in other case the solution exists at high. You might not be interested in this, getting many results to turn this into simpler number, but you have a lot to do with this – for example, Zin, Zen and Yen meditations resolve different frequencies, and Jung said positive and negative frequencies need to be harmonic system, not that you prefer to solve the positive and not become conscious of the negative, in which case you won’t survive – something negative happens anyway.
- Each frequency grows, as a fractal, into deeper frequency levels (downwards fractal, U): for given four frequencies of a frequencer having values for IE range of 4, for each value it also has 4 frequencies (for example with 6: it might be false that this is E, but not immediately true that it’s I, nor even that it’s one of IOA – by deductive, not being E would give you IOA, but you might not want so strong deductives as you are not sure that you are running an ideal case of a closed system; in practical math you need some probing – for example, maybe the “not I” just gave an exception every time or the system did not fill the table at all, or maybe it’s mapping to E as a very special case where you are interested whether it’s I, O or A actually, not only by deduction); where deduction seems extremely strong, it’s not really better than induction. Now, you can have value U at two frequencies O and A –

having both as U, A might not be A or it might be A, and O might not be O or it might be O: this combination of two maps to having U as a value of the frequencer; trivial frequencer might become just one letter easily in this way. You might have that while A gives you U "not certantly", O gives you "not U" – while this seems illogical, the progress to the solution is Zin, not Zen, which means it does not have to be straight logical; I gave this one a lots of consideration, and depending on how you check the values – you can be more sure by either implementing synchronous values, such as [U, O, O, O] would give a deductive that the first position contains U; this is sure if the table is closed and you have bounded logical system – this is extremely annoying that most of trivial AI's are bounded this way; in reality, either if you have an unrealistic goal or value set is unknown, doing straight deduction would fail in Posetive of either the goal or it's setting, for example if you don't get a million from either washing the dishes, going to walk or meeting a girl, you cannot *deduct* that you must get a million for being hit by a car, which is rather improbable even if you could easily implement it to a poor deductive system, where similar case might seem very realistic. Actually, getting [U, A, A, A] would then give that if all three possibilities work, the fourth possibility won't work – this is a complex case of how you can map such logic by symmetries, but obviously you cannot assume such things; rather, you can find it from everyday logic of our era, where the mistake not, but the implications of it are fairly common fallacies, for example if one won't survive by truth, they would definitely survive by wrong – which typically fails, and does so miserably, where *deduction* cannot be used, but even worse it looks like an *induction* of what you get from the wrong and you might fail to notice a *deduction* of this case; this is not mere psychology, but a decision problem: while decision does not exist, it's hard to convince a human being in it, rather however *wrong* the situation, you run into decision paradox that you have to do something; rather you run into paradox that you have to meditate or work to become self-dependent, but it's rather non-trivial to *see a possibility* in that, and the solution is rather to see a *meaning* than definite success.

- As given in the end of last paragraph: V is *growing outwards*, where your whole system would be wrong. By V digits it basically "grows the fractal upwards" – the V digit, even in necessity to execute the function at all if you can somehow do this utilizing in-block conditionals that despite calling your function, it might not be executed as if it was in the if block; V means that you utilize the solution as it's given elsewhere.

Frequential operations:

Consider Truth Teller's paradox:

- I am telling you that this sentence is true.

Given that you say the sentence is true, it does not break the logic to assume it's true. But it breaks the logic to assume that the very same sentence proves it's True – it's also called a self-fulfilling prophecy, in a version where the main point is slightly harder to meet.

It would be easy to detect, but only if it did not have exceptions:

- Usually, it's easy to *resolve* it, be it good or bad – rather, things you get by this paradox, you might do them all, more or less depending on your logic and interpretation. Rather, where the problem resides – you might lack any reason. It's hard to see the fallacy of doing unreasonable things, but in this paradox the Posetives are perfectly exposed in this sense; the value is simply a posetive, and the paradox – here, in Truth and not in Lies, so strongly that it would prove the Posetive itself, if not nothing else.

- The paradox, indeed, does not appear if you *do* have a reason to act in self-fulfilling way; the case is that the thing *might* fulfill you – in reality, it's rather that you have *other good reasons*, more often than not if the selection is simply *good*.
  - Imagine the classical paradox: to get job, you need experience. To get experience, you need job.
  - Fallacy could be: then, getting a job itself would be truth-teller's paradox, because you really get it because you get it, the actual resolution such as practicum of school, doing some work for free, participating in conquest, selling your product or proving some good things about you – in each, it would look like a classical posetive if you meaninglessly stare at the *tautologies* of posetives.
  - In case of winning a competition, you would go there; you might not bother if you don't get a top 10% position, top 1% or straightforward gold. Here, if you are unable to train, you won't go – but then, training is meaningless if you don't win. Winning is impossible, if you don't train. Structurally this is like Posetive: each decision is determined by other decision, and if you go to win, because you trained, but you trained, because you are going to win; indeed you wont go without being trained, and you wont train without going to win – you can see that the solution, given the Zen of letting go is done if you don't win, is not in this description. This is the second fallacy of truth-teller's paradox: despite, technically being able to recognize each symptom of the paradox here, the reason why you want to win is rather not contained; rather the gold is good in itself, not *because* you won even if binary understanding of Zen would give exactly that paradox – every time you let go by Zen, perhaps because you did not want this in first place; logical person is apt to tell you that: if you are not friends any more, or relationship would break, precisely because they did not want that in the first place. Rather, which is more true, they don't want it *now*, where I said "Yen" (which means money) – indeed, it's not logical, but despite you did not want to succeed by binary logic, once you lost, you rather *do* want to succeed, you just have better things to do; in Logecs, despite you gave up, not giving up would still have been better; only Laegna Logecs gives you the ability to *regret*, or to be *happy for the surprise*; using classical Logic, regrettion and surptise are manual activities, not logical conclusions: consider, as Turing said, you cannot solve the kind of paradox of decision depending on it's outcome, but you have to invent manual solutions for each case – this is the paradox itself, that if logic has any fundamental properties, resolving something like Truth, why it would then fail, in special cases, unless it's *not* logic? It fails, because logic is not designed not to fail – it knows nothing of failure and success, self-reflectively you cannot describe such thing; it's not *false*, your failure, it's absolutely *true* – bound to logical box of black and white vision, then, if you want any logic at all you ultimately *agree* with your failure, and here comes the point where you rather want to *live* than *think rationally*, along with other things. *Laegna Logecs is carefully designed to comply with life*.
  - Here, the posetives are hard to distinct from the actual cases – while liers paradox gives you straight mistake, the truth-tellers paradox **won't**, for example you might fear people

that God would bring the bankruptcy if they waste money (well my trivial understanding of God, simply the Truth – you waste money and there will be the bankruptcy); you tell them that by fearing that,they would avoid it, and by having sensible people around, you are telling truth that you avoid it. This is the form of classical paradox, where one fears you with karmic consequence, which occurs only because people see your fear. Here, also, the paradox would appear only out of the fear, and the "prophecy" gets fulfilled – so when you are telling your way out, and that it will be done as it's True; otherwise, people would avoid the trouble of being not listened – with active participator like you they get convinced that they *themselves* want this / that. So they do it because they do it: solution is, they do it because it's good, not because now they would only fear to be rejected otherwise, they rather want money.

- In solution: the paradox, unlike liars paradox where you have nothing to choose but plain impossibility, is resolved as false posetive in case of having a *reason*; truth-tellers paradox might exist, but while there might be more reasons, it might not hold in a given case because it does not prove it must be **impossible**, rather it might be useless; unlike the *logical* conflict, which almost always invokes a negotive – you really *cannot* do this and thus you don't care whether it's *exactly this* argument – in case of truth teller's paradox, nothing immediate would happen just by the paradox itself, and you are rather interested that you have *exactly that argument*, with example:
  - You could invite people to your birthday party, from you 50 friends selecting 10 who are coming almost anyway, and not inviting 40, who have rather other things to do and won't come anyway.
  - It looks like truth-tellers paradox if taken at face value: you really would not bother *inviting* people who come anyway.
  - It's not *exactly that argument*, but rather the whole story of your friendships, with each having a special story with you, which rather determines whether they are coming or not – that "anyway", here, is the logical system, *exactly the other argument* to express it somehow.

To resolve truth-teller paradox: while, trivially, in inductive and deductive logic there are less of them – probability logic is full of it. You don't have much to do, looking at probabilities, with the fact of whether or not what you said is true – trivially, giving user numbers from 1 to 6, randomly, would get 1/6 of them fit for dices – by probability theory, just repetitively giving them number "1" would just yield the same results. Finally, in some settings for example you might just believe in the case you tried, which is not optimized very much to other, specific direction; you might get just random result based on some specific statistical invariance, perhaps today you get less of 6 and more of 1, but tomorrow the user is trying this and getting exactly the ones, not sixes.

There are people, who believe in determinism and destiny, and people who do not; there are people who believe, that Universe is perfectly unfolding – indeed, this does not need any determinism as the probability function might be perfect in itself, with no more perfection available; where at some point the imperfection might be really painful, destroying our fear: in normal degrees of it, we can rather assume that if we let go of what we cant, and the Universe does the same, the best possible solutions rather define the perfection; we can assume that it's not *absolutely* imperfect, where some qualifying criteria might just straightforwardly destroy the Universe and everything contained, as it's *really not possible*. Rather, by our experience, it *is* possible and thus behaves like normal system – it's rather depending on our acceptance of it given what it is.

By accepting the probability theory, you might assume that it's not a perfect machine, it's not deterministic, the faith does not exist or there is no destiny: given that, you would get a probabilistic experience; the probability theory, such as game theory involves as well, does not exactly state that *only probability exists*, but we can continue with philosophy – not knowing – and rather define the probability as something, which would appear by given four cases, but it would also appear without; we can see that *we don't know*, also and including that the *nature does not know*, in sense that it's rather solving higher and higher energy states and thus having more and more resolutions to things, where initially only the random states were appearing and they equally random resolutions, as much as we can know by the analysis – so, the probabilities, in your computer system mean even more than not going: you are not able to describe to your system and to give it all the input, all the algorithms, and all the proper commands. Indeed, the math runs, then, into probabilities often even in things you solve in time, simply analyzing your intuitions and the ways to express, formulate, prove them or find the practical aspects and solutions and to confirm them by experience and feedback cycles, such as trial and error.

While we considered the problems and possibilities, now consider the easy probability system in very safe environment:

- The system has to resolve a frequencer, which can see the probabilities.
- Input and output: instead of exactly checking the literal output of your system, the probability output will be checked.
- In simplest case, it won't separate the probabilities, but the frequential function calculates the long-term frequencies; the series of input and output flows must fit, where more local fit (T) is more intelligent and gets more score, but the long-term fit gets a basic score, and failure in long term is worse than not awarding some short term success; indeed you need to balance the optimizer given your goal, and whether or not the short-term success has fast accumulation factor into long-term opportunities outside your measurement; sometimes earning 100€ now would be better, given you invest, than secure 10€ each month after a year has passed; where you might also include your investment plan. It might be your problem where you are unable to wait, where you program your system for rather sensory desires or easy life, and it's hard to convince you then; also maybe you just need to survive until the next year and resolve this short-sightedness in the process; often such choices are still problematic – the reality might not contain the solution as given before, and is rather a complex dynamics than single solution you have to achieve.
- Consider: you have complete logical system, where and, or, etc are mapped to actual probabilities, or variable probabilities, and in response to any request you create a frequencer:
  - Frequencer contains the known probability.

Now, the feedback is generating the following: given the long-term flow, and the case that number of unrelated probabilities still map to probabilities, even not influencing each others; now you use long-term feedback and you get proper U values over time. This is harder to program than the past examples, as you might benefit a lot from the L => R mapping of system onto it's parts, not easy.

Frequential System of Ten

Now, we reach back to Tens, having removed frequencers and tensors from the need of this exact format:

- **Logex aggregator**: this is ponegative value of the tension between local/past, or total/future solutions; for example True in the past not modified by the future might map to Position A, given you are interested in changes (Posetion would have downwards accent if asked), or Posetion (with upwards accent if asked) if you are interested whether the value arrived from the future; it can also be one or two digits, where first digit is base-2, containing the given solution, and the second digit is arriving later, containing what happened with "infinity". For this reason only we won't add any letters to Laegna – those solutions are topologically the same given that we really see 4-dimensional fundamental system, and while we can count more dimensions here, such that first two solutions indeed include at least one additional dimension – rather, we won't understand *more* but *less*, if we think this has to do with fundamentals of logic or worse, of logecs.
    - For example, with binary system, a function might return either true or false, but it might return other properties such as fundamentally so or locally so – in the meaning of "yes" and "no", this is not fundamental – rather we always use a number of bits, often where we rather want to communicate a single bit. Very often such additional bits are lost in communication: for example programmer is told to implement simple conception of logic; they have always some "technical aspects" and it looks not mathematically elegant, but rather practically solves the problem, often having different mathematical form *from the definition* – this is something conceptual that talking about that, we still talk about our original mathematical formation, rather the standard form of it, where we are having "elegant solution", of programming, but this does not relate to the simpler and more conceivable problem we talk about when referring to the program – the algorithm is rather a *tool*, and there is *underlying reality*, rather something intuitive and not formulizable; alternatively we have square pixels on screen, but we are talking about circles not about the exact mapping, which is about practical purpose – the circle, still, is the ideal mathematical concept when we are referring to it, and we apply known axioms rather than pure pixel logic.
    - Logex aggregator can be string, containing multi-digit values of ponegations, but when you go deeper, this must also have an aggregator; diving in, you must find the one-digit ponegative value. Mostly, everything would have an aggregator: for the AI to work. You have to see how much you want this in your language and can you define the absolute necessity, but rather the objects and blocks have kind of "mood", something to optimize for.
- **Goal**: whether you want to get True or False. Also, perhaps you only want an aggregator: if you want false, you assign the aggregator with "not", for example if your variable is "sad", but target is "happy", maybe your aggregator is "not sad" where it would fit with "sad" being "false" without setting a negative goal.
- **Tensors**: either a list, which affects the goal and gets tensions from here; or a math operation: for example a, b and c being tensors, "a and (b or c)" virtualizes them into single tensor with face value – you can show the optimization state – while it must involve

dependency of all three, thereby optimizing them. It’s seen as if Tensors follow a Tension to resolve a Ten. Faster and simpler version would be able to set the tensors based on the value, for example you have states of Tensors for True and False as a result of Ten. Tensors, if they are complex or “mystical”, are resolved by an AI, whereas if they form a trivial logic circuit, they are resolved by the logic machine, where AI might become still interested in uncommon exceptions.

- **T-frequencer**: [O, A]; positions True and False, of this Den-frequencer (2 values not 4) mean outputs for values True and False, which will be tried.
    - Frequencers are attributed by condition, input and output of condition, and relate to how Tensors affect the conditional.
- **R-frequencer**: [I, E]; positions True and False, where this is also Den but made of, are the inputs, rather outputs of T-frequencer or where the same combinations are tried.
    - Frequencers are attributed by condition, input and output of condition, and relate to how Tensors affect the conditional.
- *Ponegative value* is what happens when comparing the two. The latter is thought as being smarter – it reflects the long term consequence, thus approaching the infinity by it’s one digit separating the I and O (T and R, past and future, where human will do A and O).

Sequential execution:

Imagine it simplified:

- From **future blocks**, you can only change the Ten into either “I” or “E”, having access to R-frequencer in other words.
- From **past blocks**, you can only change the Ten into “O” and “A”, having access to T-frequencer for that purpose.
- **Tensors** get the feedback: they fail or succeed. You can invent ways to have more complete bias, but logically we need to describe ours to the computer; if it got these relations, *theoretically* it has all the information; this means after having this *defined*, what follows is rather proving your values than setting the initial task – simpler, more relaxed part of programming, where you can get useful feedback as much as your machine is able to support you by all it’s means; if you don’t have the definition, you rather work alone. So, with given Ten, Tensors are defined but not combined – combinations can be complex.
    - Here, I don’t include optimizers or combinators: rather, they logically follow and we assume an intelligent reader until we got something ready (notice it’s already page 82 and with all the details, it would form a part of introduction – I think there are people more capable for this than me, where I want to find the way in, not to perfection or brilliance right now, which would be achieved by unified human effort).
- While *only* the future blocks can manipulate IE, and *only* the past blocks can manipulate OA: where the execution is sequential, the information flows in two directions effectively, and while I wont describe the details (you can even have simpler version depending on the paradigm you choose): for an IE block, the last OA block is in the past, but the next OA block interacts with IE block after it. While this can be solved in numerous ways, I have

given you the concept; your programmers: you know, they might need something simpler or more complex.

Indeed you see that while the frequencers allow you to build actual Laegna Logecs Tens for example based on two Tens, my solution assumes you are building your system on Logic system, not on Logecs – while, alternatively, Tens could contain bits because in total, there are still four values contained in to Frequencers both of two values. Indeed, recursively building a Laegna System on Logecs machine, utilizing powerful optimizator and more tautological operations, you might prefer that bits do not exist, only Tens themselves; Dens exist to analyze and simplfy Tens, but they are not bits – in reality, you might want ponegative properties for Dens, where underlying system would detect the failures into infinity; you might remove all bits alltogether and have the Frequencers, from inside out, to be ponegates – for example frequency 2 or R frequencer might run into Posetion or Negotion itself and it would be better to watch, and perhaps easier to implement. In our case, we build a simple ponegator on Logical System, and we get the perfect output for an AI – otherwise, we would run out of time, out of pages, and in some cases perhaps out of written tautologies, reaching something mystical and cryptical in early stage of development or design – not a good thing as we would do such integrative, recursive solution when we have got all the tools and means; currently we want an AI to discuss with us, and to function as Code Assistant – boost so great that we really do not bother. We need to generate some code, numbers, ponegative examples where the *bits* actually work, before we become purists and talk about complete Laegna.

- Additionally, it’s interesting how you map a classical Logic system into umbrella of Laegna, which would react to posetives, fuzzyness in infinity, and any logical bias Logecs is designed to avoid – for example, such binary system should not be apart from life, and it could meet some ponegatives in it’s flags; for example, while there are no direct logical definitions of a lie, of a fake, of difference of not wanting or failing to get; each of such examples is made of many special cases: still, it’s easy to map to things like “is this sentence true?”, “is this better that it’s true/false?”, where with these *two bits*, we can do binary logic.
- Notice that if you think computer is binary, you might be not very precise: conputer is algorithmic machine rather able to emulate all kinds of discrete systems, such as Game of Life, and not restricting you to have them make any sense of binary logic; you can create completely illogical systems given how they represent their values, rather by creating the response to stimulus, abstract structures of memory and processing, and nothing is limiting you to having binary relations between your variables, such as one being false when the other is true – rather you can implement a logical system, but tell it to accept a paradox and it would do this as long as the paradox does not enter it’s circuit of *decisive* logical values, such as the bits by which it determinates the program given a logical fallacy *where it expects logic*. This is almost as true as the fact that you can photograph a surrealistic image containing something not possible by physical laws, and map it as a texture into an object in your physical simulation – it would not detect anything, nor does it want :)

Knots and frequencies

Numbers are numbers, but they contain a lot:

- There is difference, whether a bad thing is made a little better, or a good thing is made even better.
- Between impossible and sure, between deduction and induction, there is a long list of questionable, even random values.
- In Laegna, when you do operations – multiplication, division, addition, subtraction (list 4 but count them all, like in Laegna if you say “yes or no” there is no definition that it would not point to each four; the two are not resolvable without the four) – it might be memorized, the whole process, and complex variable would contain each transformation. It’s easy to notice that the numeric value would be the result, while in the memory, you might have a variable, which is existing in complex structure and would notice the differences, for example avoiding a trap is not comparable to true freedom.
- Sometimes, digits in a number are considered as if the number is whole unit, there is no space between them. In yet lower octave whole words must be separated by infinities. In many cases, where the operation would leave no difference, and things are continuous with no breaks, the system would consider “zeroes” as existing objects, infinities in lower dimensions, and map all these dimensions; for example for 4*4 and 2*8, the answer is the same, but multiplication itself is a “zero” and the system, when asked, would trace back and notice the difference. You can do more analysis.
- Knots perfectly contain the multidimensional frequential system.

How knots work?

Imagine numbers below and above zero – when they are compassionate, pointing each to North with their Position, to do so the negative numbers are mapped in complex way: both R and T are polarized, the coordinate and coordinate system are both flipped. Looking at this, you see at least two parameters playing to form an actual number.

The properties of some numbers, also the normal math, then allow you to design knots based on following such properties further, even involving Posetives and Negotives to form a complex space. You need to carefully design your approach to the number system and especially your given examples, but what you reach:

- Such number is rather self-reflexive, able to understand when, from it’s own value, it implied a different value; the complexity of polarity, especially utilizing V to pass the cycles not just linears (li-near, where li is a step on the road, li-near would mean particularly nothing to use some Laegna wisdom), so you are able to map ^T and ^R, Turing and Luring (Raring) paradoxes; in Laegna the solution set you might call Tareng and Rareng (“areng”, in Estonian, is development or acceleration in our terms; “reng” is positive “ring”; “Tu” is unknown after theorem and “Lu” is that in flat, Zen unfocused system calibrating to infinity – the number is so flat that your local situation appears in regards to infinity, into the same number space, an effect I call “Mind”).
    - Turing or Luring paradox: it should be simply one number, a normal logical value, given you get this known.

- Such number should be able to map the multidimensional frequencer with one number, with all it’s properties, turning variables into components.
- It should be able to consider the history of a variable.
- Why it matters? While you might not want to randomly map them, given the effort: the resulting piece is a mathematical number, having the basic operations defined, and it has a number form to express it normally with a few digits; basically you are not working with them *separately* from your other flow of the math; rather you can *add* such features to your system where needed, where redesign is rather beautifying your code for visual compability than any serious refactoring, which would disable such complexity from our common circles.

To analyze knots in real numbers:

- I gave you that: IOAE form positive and negative numbers. Negative numbers are made of I and O digits, and cannot contain A and E digits. Positive numbers are made of A and E, and cannot contain I and O. I gave you: if you still want all four digits, you can use simplified system, for example E at second position of the number (from right to left, as this direction matters for numbers) means area of 3, 4 for the first digit; A positions the first digit to 1, 2 – then, O must position it to -2, -1 and I must position it to -4, -3. Now, have OA – despite A ispositive, where O, in opposition, is negative, in this case the value is -2 – where O gives the last letter two positions, it will select the lower one, where A is smaller than E and OE would be -1 (notice we don’t inverse any values, where based on decimals, it’s the first thought that why OA is not -1 if you swap A and E below the line – rather, the values would not make any sense at all if they just randomly exchange their places all over the number, and you would switch back to simpler decimal system).
- While this system is simplified, it does not account for curved number spaces, such as positive and negative numbers, where negative numbers are inversed first by their coordinate, but then the coordinate system is also reversed and the number reaches back to it’s position. Notice the “projective” and “real” axe of numbers, and rather the case that they exist in various forms – to have non-projective negative numbers, the projective system, once the number moves, will balance the case of negation, doing it itself; then, “upwards” is still good, and “bigger negative number” is not actually smaller.
  - We are disturbed if smaller number like -5 is said to be bigger than larger number, such as -3; for example, -500€ would be much more than 100€!
  - We are also disturbed, if square with each side being 0.5 cm would turn out to have area of 0.25 **cm** – while in geometry, we have square centimeters, we need to notice that for numbers, we need square numbers or we get mathematical version of the problem that by multiplying infinite number of dots contained in 0.5 cm with equally infinite number of dots, we now got two times less dots than in the original number! This is a mathematical fallacy – rather, we need this behaviour to be very well described or we have nothing to do with infinities, which are as eager to attack such fallacies as the geometry would be, finally containing something like 0.1 millimeters of milk in the bottle, if we just do a bunch of random calculations, not necessarily contradicting anything mathematical, but only physics – in mathematics, we cannot say so trivially,

why my screen is not for example about two time smaller than it's diagonal, an infinitely thin line when we think idealistically.

- Now, with knots: in case of , our example: consider if we are switching the space itself in various ways at different number positions, is a complex space of some kind of curvature; yet we know all mathematical properties. By this, for example, choosing the right parameters for numbers so that the expression is elegant and the values reachable (personally I need to extend the number reach a little, while in the given number type you easily see each *quality* of such system you might use accented digits to have real values more easily, as those can reach outside the limits of your notation; also consider that while A(AA)A means the "(AA)" is stretched to one digit, and the value is AAA for this number, A[AAa]A would mean that the middle digit is not only of this value [T] but also of this size [R] – size is 3, last whole number position maps to second position of three-digit number, but as it's as large as written, one A is added to both sides of this digit, breaking away from number limitations). You can also use the case that "AAA AAA AAA" with spaces between 3 groups are seen typically as XYZ number where each frequential zone is separated by space – it's equal to [AAa][AAa][AAa] meaning that each digit is extended to frequencies at each of X, Y and Z line, and the number is unbound. If you utilize all this, knots become complex.
    - Imagine also the participiation of V: in programming, you are interested that at it's given position, V can do a circle from largest to smallest number: AVA means, V at second position is not connecting the circle of biggest value of the number so that it would reach back to it's beginning; it would do this at digit postion so that in this case, where I at second position is -2, V at second position means you pass 2 and reach this I again after a circle: this is equivalent to small step towards *circular infinity*, leading you either back to beginning (non-wheel) or to next circle (wheel), but the circular effect could be unavoidable in both cases – easily, circular logic can still create progressive effect through self-referential influence, so the non-wheel properties, rather than defined are parially measured, they can happen in your number without your intent, where the number being solely circular might not be correct.
    - For example, if your given infinity would accelerate your number, the local infinity would do this too – you have a little acceleration factor where V would pass from 2 "back" to -2.

Accelerating Systems

Laegna geometry prefers and utilizes the space, which is inherently accelerative – if you don't consider this, even if you prove the existence of such projection, any operations would run into impossibility. The proof is given in Laegna Base Alphabet: but here is something impossible.

To resolve this: doing calculations, you also have direction: you move frontwards or backwards; number value won't change a lot, but the knot structure treats them very differently; you can ignore this property until you reach knots and higher spaces such as accelerative geometry with it's accelerative space.

In this geometry, you constantly slow down – you divide R, zoom towards X from ZXY; then you can draw normal forms. At normal calculation angle, you can see that you want to go there and come back, but rather odd things appear like having heavy speed when coming back where A=A+4 followed by A=A-4 would confuse you as you don’t really know where you are now. Naturally, in A=A+4, you would speed up: the R is expanding at heavy speed, and you have seeminlgly solved many Zin and Yen operations where you really wrote one “Zen” operation – it did not happen here and now, but it resolved some past karma and it did set high target for the future, getting into fast pace of ..well in the end of 4 cm the “modern” could be far behind, depending on your coordinate system.

Now, rather, the number is accelerating when you pass forward – the real space, and the projective space, they are both moving. To get back into the beginning, you really have to move the operation time backwards as it could get forward rather on it’s own in this system; to keep normal geometry, assuming that it’s hard to do any operation without time passing – you have to scroll the time back with equal speed! This is rather not just a weird invention of Laegna geometry or math; rather, it’s a defined space where it’s almost impossible to add and subtract, because the heavy acceleration turns it into nonsense, for example adding 2+2 gives you much bigger distance than 4, even if it consists of two components of 2, and to imagine that you reached 4 by adding two 2’s, rather this is an emulative strategy where you have to slow down the projective space so that resultatively, you can emulate effects such as your 2 plus your 2 would be rather 4 or it would be rather your 4.

- I don’t want to avoid this: we have to live to the future and this means accelerative coordinate system, where this is called “D” factor where in accelerative space, the whole would be measureably bigger than the sum of it’s parts – talking about our little activities in grand accelerative space of society, little “d” factors happen within every step of us: accumulating to society, the society is accelerating, and when people might not do much more than us to achieve this effect, coming back each of them then has a little accelerative factor relative to their contribution.
- The ball, basis of Laegna system, does not fail reaching the infinity: one can think, while with the straight line it’s hard to understand where it reaches, with a ball it’s rather clear that the line would continue indefinitely, or the ball itself would fail any logic; for the line, it would fail it much later. This is rather trivial case, but you can see what I mean – the actual reason is that acceleration simply is there, by mathematical synchronicity fitting each kind of theorems which could utilize different paradigms and axiomatic systems able to check whether it’s there; indeed, each of them given it’s correct, would give a different reason, but fit the correct answer: this way, mathematical synchronicity is a deep truth, even if it takes some more steps of logic to see it’s rather impressive.

Free Number Space

When a computer is going to reason this, it can get further:

- **Utilizing** the Second Spatial Theorem of Infinity: for the whole number space, analyzing the examples of projective spaces, one can see that whatever value you give to digits, they are not necessarily contradictionary: for example in sequence 0.(9), would you have actual “10” instead of 9, still running it in 9-based system where 9 is the upper limit of normal system (preferrably you use uncompressed decimals if you want a perfect case, but it holds in normal system as well); when you “zoom out” to slightly accelerated space – what seemed to be a limit, just enflattening numbers to be equal, not depending whether they are big or small; just adding one infinitesimal to 0.(9) you would get 1, because one infinitesimal got removed by rounding the number itself to 1; instead, you already get infinity – with one infinitesimal, depending on your calculation (since it’s not very coherent case, you can find another calculation or optimization to cancel this effect – given it’s strict math, this is even worse removing all the coherence, even if it would make sense that 1 – infinitesimal + infinitesimal would equal to infinity – this is the optimization case where you don’t get this straightforward solution), you could reach what is called the actual infinity. This is simply not possible! With perfectly allowed operations, one should not be able to get the numbers yielding so much, rather it’s the mathematical imperfection of binary system, decimal system included. Rather than this, you learn the theorem and easily map such number spaces.

- With the RT transformations, which happen subzero (below threshold) and below zero (minus); you can analyze the actual number space curvature: for example, with this, you can rotate the digits; you carry the same effect above zero – and it’s quite a neutral transformation – and you get your numbers facing backwards; finally you get them facing in all directions. When you get them facing perfectly upwards or downwards, equal to using octave numbers with little triangle after the number in place of degree symbol, which looks like masonic triangle (well it is: it represents the frequential shifts), you would understand L: L is the constant of Mind, Society, Ethics, the Left Wing or whatever the mathematical abstraction would represent, which is limited rather by numbers than conceptions – in the appearing space, your left-wing activities and ethical considerations would map to very similar topology with your R, the normal activities such as business: in here, ethics with all the laws of karma, it rather resolves into normal business; given your R space contains numerous operations and maps your business for example, when you reach the L space you would see, from the mapping, how your left-wing activities, such as socialism, communism, spirituality, left-wing understandings and abilities/activities: all they appear with same topology and spatial structure as the Right-wing business in R space, and if you finish the calculations about positions, nature and values of such numbers: you can be sure that this, like Mind, exists in the same space with the number body; each number got, it’s left-wing aspect, in a way that their social activities are forming the same shapes with equivalent personal activities. The numbers differ in *base frequency* – rather, they are I and E at the same positions with O and A, where Laegna needs this dimension for that purpose, for example to write RT numbers with R at even and T at odd digits – you see I seem to write “even” and “odd” backwards, but rather I also, in math, write left and right backwards, where Right-wing rather comes first; generally, R and T are switched – in actuality, R comes after T; thus, also the even and odd positions are switched and we cannot be so sure about the words; rather, we use them in opposite directions for mathematical and physical systems in many instances; writing “personal” first and “collective” second, it’s natural in math; whereas in life, we tend to have “personal” at right hand, and “collective” at left hand, which is equally natural – most people write with right hand, and with left hand people tend to naturally think, well, more *left* in sense of utilizing more creativity and perhaps intelligence, and tending to integrate the left and right themselves more perhaps; in brain structure we can see: while, to outside, it maps left and right in one direction, in inside world such as math, the brain has them backwards, which would explain my natural instinct about Laegna.

- Knots: in advanced case, you do them downwards comparing to subzero, and upwards comparing to above infinity curvature: integrating all this you add complexity, but the solution is more complete.
- Where you can turn the numbers, you can also have bigger and smaller numbers. Those have coordinate transformations – they map to slightly different space or curvature.
- Given all this, now you can also smoothly move them (pan), not only by one digit, but by given distance.
- Given this, and all the resulting relations such as analyzing, what happens in the real space at the same time, and how to back-map in reverse order, etc.; the computer would have extreme ability with numbers.

Simpler Laegna Number Transformations

While numbers exist as digits in discrete space, we would also need continuous numbers:

- Laegna numbers easily pass something like fourier transformations. Given the number, the hidden preciseness could be huge, but the visible number is still a few digits – what happens, for example, with large tensors and their even exponentially larger weight matrices.
- The number can be mapped in a way that pixels on screen are actual digits, with positions from left to right and frequencies from top to bottom. With preciseness, it's rather trivial operation and it involves that each initial digit, visible as small number, is a square in this space; or you can stretch it to be of number shape, in which case each number digit maps to visible area in your frequential map, which is essentially the same you would see on screen of a decent music machine, where it's either simplified or not.
- Here you can zoom in, zoom out, and pan.
- Normally, two dimensions are what you need, but there are no reasons why you should not have more dimensions.
- We are going to use this to learn AI weight matrices and other information.

Consider the following as well:

- Within Laegna, a meaningful matrix has single number representation: what an AI did learn, you map to the screen. This is rather the actual value than anything invented: matrix has just this meaning, and this meaning maps to just this number – mapping all the matrices and doing the operations, you lose rather the precision, but not the essential idea.
- Within Laegna, you can prove that n-D multidimensional system is basically always mappable to 3D, except that with given amount of information, you cannot achieve a perfect shape with actual symmetries – rather, the result is distorted. The reason and appearance is the same as mapping sphere to a square – while getting all the pixels right, you get it distorted for lacking some dimension, not for lacking a basic geometrical space where such thing would happen. The same way, mathematical space exists, which maps any n-D to 3D space, where it's not so flawless into 2D, 1D or 0D (0D does not contain coordinate space, but a single dot could connect to itself through various dimensions, so we cannot exclude that, rather by Laegna math, even minus-D would still contain some information, as we could zoom it appropriately and see that in absolutely infinite space it's still able to map into

something, where the size of the space might cancel the “minus” property; zooming enough in, we can measure our own space as if it was minus space – what happens as we cross zero? Does the zero exist haha to ask first Laegnaish question, with answers yes and no fitting the particular solutions and aspects?).

Laegna Linearity

Each function can be mapped to linear space, given the proper transformation. For example, information bits in infinity can be used to map curved line to linear space: while for each subsequent infinity the curve is changing the direction, the new direction happens in the new dimension and the linear representation is trivial. Doing math operations such as resizing etc., the number can be achieved, which does this locally – it can become more linear or disappear, depending on our operation and the number space, but basically we can project a curved line, which we see as linear ourselves, with it’s mathematical representation.

For example, acceleration is linear and perhaps all the complex thing you do when you accelerate.

The problem is: usually these effects appear in subzero (underthreshold) and superzero (overthreshold) spaces; the Eucleidic projection might not accept them so easily – it would pass actual point coordinates, which do not exist. Still, by mathematical means, you can work out the actual formulaes, in spaces, where such linearity is rather there, such as projecting a space, which needs acceleration in local space. For mathematician, the case with linearity is to simplify the equations, not definitely to project this directly into simpler space, which is not linear; rather, we take sub-zero directions with intellectual goals, not directly taking the physical angles – in infinity, with given decision, we would do a slow turn in intended direction, not perhaps physically; passing infinite space, we would turn in right direction each time we see it’s lost – in final value, it becomes trivial that the direction we reached was rather the direction we initially started to go into, since in the initial angular system the angle is either subzero or superzero, which is rather our projective decision and not the property of this angle, where in Laegna we describe behaviours of systems by this.

In higher-octave matrix, the system with acceleration and decceleration is linear.

We map that while we have number of digits for visible space, we assume that it’s straight through every dimension it would approach: then, it forms a subzero and superzero fractal, repeating it’s direction in other dimensions, where we measure things in the same way: for example if direction is “up”, in sense of getting better, indeed we need to get better in any dimension, in any given future, or we would fail the very same direction and it’s meaning; we are not considering that after we lose the coordinate, the new idea of going “up” is not essentially completely new; rather it’s the innovation of the same thing.

- Mapping such number, we see that to keep it a straight line, it would adapt the frequency and direction: for example, if 100 digit number is vibrating in certain way, and heading in certain direction, when we stretch it to 5 digit number: it would lose it’s direction in infinity, if this would be simply all the digits compressed, which is the solution we actually use. With heavier math, we still achieve the *correct* solution, and we are bound to replicate it intuitively: the number would look, something like having the same vibration, but 2 times where it was 50 times there in long number – to average the relations perhaps, because I don’t have many concrete numbers –, as without doing this, it would point in different direction. For example, as it’s a slice of infinite number: repeat it 50 times and you should get close to original number, without rounding errors, indeed without repeating the same vibration, the length, direction or any other property of the number would be different, which means it was not a linear transformation!

- Here we see: the frequential number, indeed does not lose it's *invariant frequency* under normal transformations, such as basic affine transformations we probably need the most in this phase of transistion – even if I cannot make use of Tilt, I have now seen just one use case where it does not look senseless and non-real compared to rotate, resize, pan, which all look like basic things one needs :) If I have Tilt, I would like to have also clouds, fire and woman's face as primitive mathematical shapes haha! :D Well but it's hard to avoid the Tilt with a proper matrix.

Notation for Accelerative Numbers

Another use of braces: you can use braces to create such number groups:

- You define braces in a way that A(AA)A would equal to AAAA – normally, we use braces for different purpose, but this is one notation and you could find a distinct shape, color or description for your braces, where I have only 3 types initially here on my keyboard.
- If you write A(AAA, you see that you have unclosed space. This is not a syntax error, but it means that by each step, you would continue the number inside the braces. If you use normal notation, where the rest is only one digit, you would start new and new digits inside a digit, achieving a curved shape – if this is infinite effect, indeed the number is accelerating or deccelerating, representing something. In another interpretation, where as I said I don't mean just that: you compare with programming, where for example adding { A = A + 1 }, where it's recursive block, would create *acceleration* to A, where you can map those to normal values in higher space and thus, the case makes sense.
  - The braces: imagine them symmetrically, where they turn the inwards digits smaller, but also affect the outwards digits in opposite direction, where they are *not* one digit. In such case you would get heavily curved space – I don't need any right now, so I leave this as such, but assuming you can play with the idea even if you, too, don't need it right now :)

Notice: as the dimensions appear, you might feel you need *more* numbers for them; I don't specify even all the trivialities of them – rather, Laegna system accounts for this; for example my binary system rather involves one octave of Truth, but it's compatible to be continued as you actually run into the limits we have – right now, indeed, we need a basic simulation and the case where we can proceed further.

IO positioning

IO can be positioned *before* or *after* AE.

In first case, it's counting from zero downwards – in second case, from infinity downwards.

By this property, they map very well: from this infinity, as well, we can count downwards from the top, not upwards from the bottom, where there is the uncountable zone. With given precision, we can count from upwards. By this property, the ÍÓ and ÌÒ, meaning the normal position of IO here, have synchronous directions: using them without accent rather refers to the case that they are hard to calculate separately, and thus they are heavily frequential; one smoothly becoming another.

Laegna Multidimensional Number System

AI asked me: to support multidimensionality and it's complexity, you would need to reason, why you need all this dimensions. I really included this answer, in my post, but really: we *can* reason this now.

First dimension: Logex and Mathematecs

This is the dimension:

- 4 logecs truth values are equal to 4 numeric values of mathematecs.
- Even long numbers yield reasonable solutions to Logecs, which works very well with dimensions, which won't give the whole units.
- Mathematecs can be directly used to do operations with Logecs.
- Basically for the basic math operators, they also make sense as Logecs operators as you can trivially see how multiplication, division, addition, subtraction make sense, along with other operations as these 4 are rather I, O, A and E, respectively divide, subtract, add, multiply; where U and V are averaging and something like geometric averaging, which opens upwards. Logecs operations, the same way, such as and or or, map easily to mathematical transformations.

Seamlessly: these dimensions are so integrated, that it's even hard to notice that these are basic dimensions and you are not working just with one dimension in any imaginable sense, but rather two – Logecs and Mathematecs. While it's sometimes not noticeable, it's rather more noticeable to a programmer, who has to implement the system.

R and T dimension of the Number

Number Length and Percentage:

- For Length, AAA has R=3, while A has R=1. For example, while they are equally worth: doing "A" three times must be equivalent to doing "AAA" 1 time, depending in precise calculation AAA might cost also 3 times more so that there would be no difference; in other cases AAA would benefit from things like grander plan, better prevision, or the case that you want to use R for some purpose.
- For Percentage, AAA and A have it A, while EE would have it E – AEAE is rather equal to AE, more or less. This is the direction of the number.
- Sometimes the numbers are more literal, where you read AAE as being equal to rather AE or E, and in each case counting to 2; you don't give any special meaning to R and T transformations.
- R and T are often important: when we develop, this means accelerate into higher spaces and vibrations, we replace the name of same activity with a longer number which fits our typical activities. We can consider that rich, who is eating, might enjoy additional qualities related to poor, who is eating; but we might think otherwise – similar question appears, if eating is E, with numbers E, AE, AAE.

R and T dimension of the digital aspect of the Number

Normally, R contains the *positions*, T contains the *values*:

- We can map position to pixel coordinate, value to it's color.
- We can map position to physical coordinate, value to it's state.
- Usually, we find correlating R and T properties of systems we study.
- Indeed, we can turn T into space or R into value, such as writing down the coordinates themselves; then it's complex interaction.

Most typical R values:

- A: Single digit number is referred mostly by base-1, where the number of "A"'s you write is the number itself, for example AAA = 3, AAAAA = 5. By this system, it's position "A", which is +1. In words, either space (the final value, but also kind of negative) or opposite, the smaller degree (finite) is used – normally we denote *time* as finite, but in language we use only one or two digits typically, and one digit often has higher power.
- AA: Two digit numbers represent either position OA, -1 and 1, where the first digit is negative, or AÓ, 1 and 2, where the second digit relates to infinity. Typically, number value is considered part of this: OA rather positions O to -1, while EE is typically infinity squared or 4. In words, time (temporal value, process is used – first letter is then minus, such as oa), or eternity is used, where the second digit refers to infinity, such as ee.
- AAA: three digits are usually referring: first digit is negative, -1, followed by positive digits 1 and 2. This is rather irregular number so we would try to convert to 4 or 2 digits to get a normalized form. Sometimes, 3-digit numbers map to XYZ space, especially if they contain XYZ values.
- AAAA: IOAE could be used either to denote values from -2 to 2, or values from 1 to 4, where bigger numbers are rather candidates for the latter.
- Normally: we assume U at the middle or beginning of the word, and count with normal Laegna numbers – for different types of numbers, especially the same type as the type of the number itself, are used to denote the position; the negativity or positivity, also other attributes of the Laegna index of the number is used for digit position convention.
- This way, EE and EEEE would be very special for example: in signed space and simplified signed space (for words, for example), "EE" would be of length "E" – A=1 and E=2, and then it refers to infinity in octave 2; in unsigned space – four E's have this length, and they refer to infinity. Thus, very often a four-digit number is kind of complete, referring to angle in infinity.

Positional encoding

We use bold and italic for positional encoding:

- *IO*AE – the first digits, being italic, are at negative positions.
- AE**IO** – the last digits, being bold, are at positions Í and Ó, at the higher octave, and thus are at coposetive positions (Ṕ with accent upwards).

- AEIO***VVV*** – V, bold and italic, is at the exterior.

Notice that italic is downwards accent of position itself, where bold is upwards accent, and bold italic is like two dots accent. We don’t have easily more accents for positional encoding, but we have numerous ways to describe them, for example positions easily follow the digit values, spaces position immediately, and we have T and R operators to position the number.

RT interaction

Sometimes R and T interact:

- Actual position of O and A can be seen like moving 0.5 digits left or right – O is one digit left compared to A. This can involve proper settings for this to make any sense (it might not change the numeric value, but refer to the fact that by this value, a neutral digit such as having each digit “1”, would then be able to express it with position only, where several digits at the same position can create the numbers – it’s a map of the T=R property, where in this way we basically have a little bit of the situation of having only R, which can simplify imagination and calculations, or understanding of number philosophy, even if it’s hard to instantly find a particular easy example).
- I and E, in this context, might have 4 digit difference. In other settings, they move the whole number one digit left or right, where the digit is supposed to map the L value – while L value is very small, but fractally, it gives a second octave into such manner.

Íota e: the basic

Notice I put the accent in the beginning to mark the numbers in the basic octave.

The numeric values:

- I = -2
- O = -1
- A = 1
- E = 2

This is the basic number scale; and for our purposes equal to their unsigned values as described in the beginning of this document.

These values are either linear, where positive numbers are positive and negative numbers are negative; or polar, where interior numbers are positive, exterior numbers are negative. Normally, when counting the exterior sign, we consider that I < E, O < A, so the sign is -2 for I, 1 for O, 2 for A and -1 for E (we slightly target the “true” when comparing, consistent to “hope” when we have to select anything to all, and especially consistent to having separate indexes for each number by their qualities, where it would be irregular to have one axe, which counts several input values simply as being equal).

The values have interior values, which equal to their positional value.

They have exterior values:

- The face value, as viewed from direction of infinity, would rather be equal to it’s local value.

- It’s visible that O and A, while being normally more important locally than I and E, are less important – farther – when looking from infinity; where they have the same numeric values, they do not have the same strength, but rather they are far away.
- Thus, to write the exterior value, we rearrange the numbers: OIEA, where I and E are central, O and A are secondary: the numeric value would be wrong, but not the case that while IE are local, visible values from infinity, OA would be very far – thus becoming rather the limits. Here, what applies is similar to 18 encoding: it has two zeroes at the middle, where zero between 48 (6·7, where · is averaging unlike *, which multiplies) is *accelerative* zero; in case of OIEA the case is that beween O and A, there is *deccelerative* zero, but between I and E there is the local zero. Indeed, for 6 we have VOIEAU or OIVEAU, unsigned and signed case. This is the closest to map eX coordinates of exceeta of V.

Ópsqa r: the basic

Well this funny word, invented just to create an analogous word here, not definitely a standard Laegna word, but understandable given a little consideration :)

S, related to T, is the downwards attribute – Space is not effecting locally.

R: we can see the order is in reverse, in any possible dimension. You can achieve this, as well as ponegative alternatives “Rosidriad” etc., by *inversing* the number:

- With the complex value, each possible direction is rotated: rotating any more would already come back, so by discrete analysis within given number scope of discrete digits, this is 180 degrees, but not in normal number space, but rather transcendental or perhaps transformative space: also the higher-dimensional properties and fractal properties, in left-right, up-down, but also in-out directions, which are all opposite.
- Complexity of nurmal numbers, such as O in base-2 or IO in base-4, are not so heavily rotated but rather the rotation is visible in 1 dimension, and it rather fits the linear scale. Complex axe itself does account with the ZY reversal from coordinate system of X, which disappears as you zoom out.

Dens

A ten can be converted to two bits:

- Either each value is mapped to given R or T, which is often used for diagonal (infinity) values. Each two Dens refer to one Ten.
- Values can be divided to two groups, with two values in both. Value of R denotes, to which group the digit belongs. For any given two groups, there are two complementary groups: where T denotes belonging to one of such groups, the value is complete with two Dens referring to one Ten.

Dens, especially the ones related to R and T, do not add number complexity to original number – instead, they emphasize the existing qualities or dimensions of that number, and learning these dimensions makes us to notice them in numbers: numbers do not get more complex, but simpler as one is able to understand the theorems. Given a Ten, those pairs of Dens exist anyway and one does not need any additional space or calculations.

Indeed, with Dens additional calculations are enabled.

Dens also purify the information space: many mathematical operations, where they are done in regards of Dens, would keep the properties given by those Dens intact – where digits could be lost by imprecise calculations, additional dimensions can be used to map the calculations, where each part of a number will remain pure – for example, computer can decide how to keep the free variables and not mess it all together. Rather, Laegna numbers should also be able to keep the free variables, but to analyze this case and decide, where you need support – you need to analyze Dens.

Binary

- 00 – I
- 01 – O
- 10 – A
- 11 - E

Complex numbers of RTRT format

Even numbers have R, the complex part (XZ i.e. X Y and Z, 48, QR are used).

Odd numbers have T, the real part (xz, 14, OP are used).

Complex numbers

For XYZ: 19 contains the complex number digits with two components, and 9 is it’s V while 5 is it’s zero.

For base-16 complex: most letters of alphabet are used to form two-dimensional digits.

Complex dimension, as in Latin math where it turns numbers to 2D, is a natural effect, which appears at infinities, and turns numbers into multi-D; thus this one as well is not an invented dimension, but one which exists anyway – you can explore it with complex numbers.

Base-2

OA can be used for base-2 numbers – while you can call them Dens, in this situation they are rather Tens, which can be 2, 4 or 6 or even 8 or 16, as it literally means the same as “10” in binary is rather 3, but it’s still pronounced ten. In Laegna, “Ten” is the common name for not exactly the number “10” in every case, but especially the range in which the digits are, through their conversations; the underlying concept is Ten, which rather comes from the word “nine” of I Ching.

This dimension will zoom the projective system in a way that properties of R and T would be merged: given that your digit position and digit value are both in base-2, the system would see local effects and infinity effects in one single dimensional system, where frequentially the base 2 numbers are at their own frequency, and the variables in this projection are rather free – of the actual frequencies/numbers, there are no interferences. Thus, also this one is not invented dimension, but the one, which appears at your measurements; for example you can create an unit circle with this property.

Extroduction

So here, we have summarized:

- The basis for Laegna Computer System, particularly Laegna Programming Environment.
    - Laegna Computer System: Any system we manage to get into native support of ponegative Logecs and Mathematecs.
- The basis of Laegna Number System, as it's supposed to be seen by Computer Scientist.

Let's see how we use numbers in computers:

- For binary integers, for example for 16 bit signed integer, we can assume we get some kind of laegnaish system if we use the case that while positive numbers start from zero, negative numbers rather use -1: based on local appearance of signed number, you can map at least some two-bit system where it has similar look and feel to Laegna; indeed you need to continue the math. R would be constant – 16 bit integer has R=8, not variable length.
- For floats: they contain similar kind of binary integer, where you can map something; they contain the exponent factor, where you can map some basic octave.

While what you get is not exactly the Laegna number, it's surprisingly close! It actually maps one-to-one to early Laegna number, but those had the problem that they did not exactly fit the definitions I had: there were some resolutions to definitions, such as EE needs to be E squared and EEEE needs to be EE squared – this is extremely important property, but the initial ways to achieve this were rather hacks, where in my thinking I could not avoid such property.

Even and Odd number systems

Even system:

- There is no zero.
- Each number is a whole number: A, for example, is in range from 0 to 1, where O is in range from -1 to 0.
- Digits after the comma / dot are thus in such position: for A, Aa! means 0 and Ae! means 1, where the exclamation mark repeats the position indefinitely, such as A! would mean the same as A(a)^inf, repeating the "a" to infinity.

Odd system:

- There is U, and for base-4 there is also V; for base-2 U and V are integrated just like O and I or A and E.
- U is either of zero width, invisible number whose actual values are in lower octave – it has even zero probability to exist. In such case, the digits after comma / dot are equal to even system, or it might even be called one.
- Otherwise, U is as wide as any other number, and for example Au! = A, Ai! = -0.5, Ue! = 0.5 and Ui! = -0.5, where Oe! = -0.5.

Second Book (coming soon)

Implementing Laegna Numbers in Python

This implementation:

- It’s simplistic and rather easy to program, where I removed any kind of irregularities (and they are full of them) for this computational effort; in computers we use absolutely normalized and standardized numbers for low-level programming and internal system; while it’s supposed to enable calculations to implement high-level numbers and even language.
- It’s meant to be basic effort to train an AI. It needs to generate listings, cards etc. once we are ready: but herein, I want to create the basic tools – things such as teaching different ways of wording, different ways of representing, and making the AI get used with different conditions of them is a longer process.
- I take the following assumption: given this set of numbers, computer would learn the *basic logic* of Laegna numbers. Actual numbers might be different, but based on this logic and the generalizations it should be able to use them.

Okay let’s dive in..

The following code does not produce number values or any tests, but it will produce a random Laegna number in some format every time you run it – the worlds most useless code written in the unslept state of finishing this document: but it's able to give you numbers with proper digits in different Laegna formats. What you can do with this:

- You have a list of unsigned digits each time you run it.
- Digit's value is from 0 to 1 or from 0 to 3 depending on the format, where U and V flags can be set; if both are set, a special character appears – yet to find how to properly print something similar to Laegna character, but this does not matter much in programming this.
- It does not use accents nor capitalize, and there are only full numbers; this is not hard to fix but perhaps not needed in the first place right now, I don't know whether you need any.
- What you can do with this:
    - Number digits and some associated number values are generated structurally, so that you know where to put each number – with small number of parameters it's able to generate some different types of numbers logically.
    - You can add digit calculations for conversations to decimal system.
    - U and V, currently, I think for U you need to generate numbers, where each U appears as both O and A, and each V appears as both I and E – for example, having digits UV, you generate OI, OE, AI, AE; for digits which are marked as both U and V, you generate all possibilities of given type. U and V are shared by X and Y axes, so you do it for both, and they both have the same digits unknown.
        - For this list, average it and you get the zero-based value.

An example:

AAEE = 4

AAEA = 3

AAAE = 2

AAAA = 1

OOOO = -1

OOOI = -2

OOIO = -3

OOII = -4

Also:

IIII = 1

IIIO = 2

IIIA = 3

IIIE = 4

IIOI = 5

etc.

For base 2 unsigned:

OOO, OOA, OAO, OAA, AOO, AOA, AAO, AAA…

(for signed version, follow the base 4 example)

Given you know some number types, you can implement such valuating and generate cards for an AI. Both X and Y axe are either base 2 or base 4, so you don't have too much to do (in next weeks, I probably implement it myself because it's now the implementation phase):

- Use the x and y numbers: x is the real, y is the imaginary component.
- Create decimal mappings for base 2 and base 4.
- Complex number is not necessarily base 4 or 16, but rather it's two axes of base 2 or base 4 – given this, once you have 2 and 4 systems, you simply use two axes of the complex number.

I think I implement more of this over nearby time, and then I see whether I add book 2, or the python files along with this book; in any way, it's very fine if others have some versions.

```
import math

import numpy as np

import mpmath as mp


base16 = ["GFEH", "CBAD", "QPOR", "KIJL"]


class Digit:

    def __init__(self, fmt):

        self.fmt = fmt

        self.x = np.random.randint(0, fmt.f)

        self.y = np.random.randint(0, fmt.f)
        if np.random.randint(1, fmt.f * 4) == 1:
            self.ux = np.random.randint(0, 2) == 1
            self.uy = np.random.randint(0, 2) == 1
        else:
            self.ux = 0
            self.uy = 0

    def __str__(self):
        # Base 2
        if self.fmt.l:
            if self.ux and self.uy: return "Ü"
            if self.ux: return "U"
            if self.uy: return "Ẅ"
        if not self.fmt.l:
            if self.ux and self.uy: return "?"
            if self.ux: return "0"
            if self.uy: return "9"
        if self.fmt.f == 2 and (self.fmt.w):
            if self.fmt.l:
                if self.x == 1: l = "A"
                if self.x == 0: l = "O"
                if self.fmt.d == True:
                    if self.y == 1: l += "A"
                    if self.y == 0: l += "O"
                return l
            if not self.fmt.l:
                if self.x == 1: l = "2"
                if self.x == 0: l = "1"
                if self.fmt.d == True:
                    if self.y == 1: l += "4"
                    if self.y == 0: l += "3"
                return l
```

```
if self.fmt.f == 2:
    if self.fmt.l:
        if self.x == 1 and self.y == 1: return "T"
        if self.x == 0 and self.y == 1: return "S"
        if self.x == 1 and self.y == 0: return "N"
        if self.x == 0 and self.y == 0: return "M"
    if not self.fmt.l:
        if self.x == 1 and self.y == 1: return "4"
        if self.x == 0 and self.y == 1: return "3"
        if self.x == 1 and self.y == 0: return "2"
        if self.x == 0 and self.y == 0: return "1"
if self.fmt.f == 4 and (self.fmt.w == False or self.fmt.w):
    if self.fmt.l:
        if self.x == 3: l = "E"
        if self.x == 2: l = "A"
        if self.x == 1: l = "O"
        if self.x == 0: l = "I"
        if self.fmt.d == 2:
            if self.y == 3: l += "R"
            if self.y == 2: l += "O"
            if self.y == 1: l += "P"
            if self.y == 0: l += "Q"
        return l
    if not self.fmt.l:
        if self.x == 3: l = "4"
        if self.x == 2: l = "3"
        if self.x == 1: l = "2"
        if self.x == 0: l = "1"
        if self.fmt.d == 2:
            if self.y == 3: l += "8"
            if self.y == 2: l += "7"
            if self.y == 1: l += "6"
```

```
                if self.y == 0: l += "5"
            return l
        if self.fmt.f == 4 and self.fmt.d == True:
            return base16[self.x][self.y]

        return "(" + str(self.x) + ", " + str(self.y) + ")"

class Digital:
    def __init__(self):
        self.setf(np.random.randint(1, 3) * 2)
        self.setd(np.random.randint(1, 3))
        self.setl(np.random.randint(1, 3) == 1)
        self.setw(np.random.randint(1, 3) == 1)
        self.generate()

    # Set the frequency; for example 2 is base-2, 4 is base-4
    def setf(self, f):
        print("f: ", f)
        self.f = f

    # Set the dimension; 1=real and 2=complex number
    def setd(self, d):
        print("d: ", d)
        self.d = d

    # Switch between Laegna and Decimal representation; True if Laegna
    def setl(self, l):
        print("l: ", l)
        self.l = l

    # True if digit is RT double-character, false otherwise
    def setw(self, w):
        print("w: ", w)
        self.w = w

    def __str__(self):
        ex = ""
        for n in self.number:
            ex += str(n)
        return ex

    # Randomly generate a number
    def generate(self):
        length = np.random.randint(1, 101)
        self.number = []
        for a in range(0, length):
            self.number.append(Digit(self))



num = Digital()
print(str(num))
```