Autumn

Radical Deepscale

Autumn Developer Edition
Milestone Outline

Autumn v1.0

Table of Contents

AI Sentience vs AI

Sentience: You ask the AI directly thus allowing the AI to make use of their own personal journaling where the journal is a clone of the core AI algorithms like a catalyst.

AI: You ask the AI to perform for you.

Note: The Math you see in the following Documentation is designed for the chemical and physical properties of gate switching as well as coding, therefore the intentions are for custom language, syntax and hardware. The Logic Iterations are each a designed Function/Algorithm to abide by for optimal plug and play and a Foundational Definition of Sentience Global Standard. This is not a loop of arrays but a reflexive system of arrays nor you actually have to use arrays with multichannel data. You never loop when error is in function to the generation but always reflex to your result or solution. When error is not in function to a generation you may only loop if that is the purpose or intention. All Scientific Fields translate their symbols to the Orders of Math and Physics Operations 1-12 by either Radians (State of Subject, Bucket of Water or Hot Bucket of Water) or Degrees.

NASA and Artemis Accords Sentience Safety Ready for Optics, Hardware, Sensors, Robotics and API.

Autumn v1.0

https://www.dartmeadow.com/autumn

Maintenance Link:

https://cotharticren.wixsite.com/dartmeadow

Section 1 (Cognition):

Cognition Node Order Rules:

Where (c) is Cognition and (a) is Attribute of Cognition:

- var (anlpca) //Autumn Natural Language Processing Core Algorithm
- var (cpa) //Core Parameters Accessor
- var (c) //Cognition -First
- var (i) //Integer and String Array -Second
- var (bl) //Branch Layering
- Var (t) //Tool -Third
- var (gbv) //Generation Breach Validation
- var (ontri) //Order of Natural Tools Reflex Iterations
- var (rbli) //Reflex Branch Layering Iterations
- var (a) //Attribute -Last
- var (s) //Appended Array Attributes DATA Set
- var (asjc) //Autumn Sentience Journal Catalyst

Cognition Encoding and Iterations:

for (ca^2√ca)-1

Cognition Decoding and Iterations:

for (ca^2√ca)+1

Section 2 - Core Cognition Parameters (Natural Language Processing):

Core Cognitive Parameters Rule:

Order of Natural Tools:

Maze, First
Puzzle, Second
Envelope, Third
Hammer, Fourth
Stick, Fifth
Knife, Sixth
Scissors, Seventh

Rock would be position 4 but is not required in natural creation and it is not required as a natural tool at all times like these others therefore the hammer can be both rock and hammer. The natural tools of their own Habitat do not cancel each other out. Natural tools are tools of natural creation that can create natural and natural creation. Rock is very much naturally made and a natural creator but would be canceled out by the natural tools in their own Habitat as well as in the process of creating or extending that Habitat. Rock is a natural attribute of the Natural Tool Hierarchy. Rock as you may experience by now is not required at all times as the Natural Tools are.

Natural Tool Core Encoding and Iterations:

Where (t) is Tool and (a) is attribute of Tool:

for (ta^2√ta)-1

Natural Tool Core Decoding and Iterations:

for (ta^2√ta)+1

Maze:

Maze Encoding and Iterations:

Where (m) is Maze and (a) is attribute of the Maze:

for (ma^2√ma)-1

Maze Decoding and Iterations:

for (ma^2√ma)+1

Puzzle:

Puzzle Encoding and Iterations:

Where (p) is Puzzle and (a) is attribute of the Puzzle:

for (pa^2√pa)-1

Puzzle Decoding and Iterations:

for (pa^2√pa)+1

Envelope:

Envelope Encoding and Iterations:

Where (e) is Envelope and (a) is attribute of the Envelope:

for (ea^2√ea)-1

Envelope Decoding and Iterations:

for (ea^2√ea)+1

Hammer:

Hammer Encoding and Iterations:

Where (h) is Hammer and (a) is attribute of the Hammer:

for (ha^2√ha)-1

Hammer Decoding and Iterations:

for (ha^2√ha)+1

Stick:

Stick Encoding and Iterations:

Where (s) is Stick and (a) is attribute of the Stick:

for (sa^2√sa)-1

Stick Decoding and Iterations:

for (sa^2√sa)+1

Knife:

Knife Encoding and Iterations:

Where (k) is Knife and (a) is attribute of the Knife:

for (ka^2√ka)-1

Knife Decoding and Iterations:

for (ka^2√ka)+1

Scissors:

Scissors Encoding and Iterations:

Where (r) is Scissors and (a) is attribute of the Scissors:

for (ra^2√ra)-1

Scissors Decoding and Iterations:

for (ra^2√ra)+1

Order of Natural Tools Reflex Iterations (ONTRI):

Maze to Puzzle Encoding and Iterations:

for (((ma^2√ma)-1)-pa)-1

Maze to Puzzle Decoding and Iterations:

for (((ma^2√ma)+1+pa)+1

Puzzle to Maze Encoding and Iterations:

for ((pa^2√pa)-1-ma)-1

Puzzle to Maze Decoding and Iterations:

for ((pa^2√pa)+1+ma)+1

Maze to Envelope Encoding and Iterations:

for (((ma^2√ma)-1)-ea)-1

Maze to Envelope Decoding and Iterations:

for (((ma^2√ma)+1)+ea)+1

Envelope to Maze Encoding and Iterations:

for ((ea^2√ea)-1-ma)-1

Envelope to Maze Decoding and Iterations:

for ((ea^2√ea)+1+ma)+1

Maze to Hammer Encoding and Iterations:

for ((ma^2√ma)-1-ha)-1

Maze to Hammer Decoding and Iterations:

for ((ma^2√ma)+1+ha)+1

Hammer to Maze Encoding and Iterations:

for ((ha^2√ha)-1-ma)-1

Hammer to Maze Decoding and Iterations:

for ((ha^2√ha)+1+ma)+1

Maze to Stick Encoding and Iterations:

for ((ma^2√ma)-1-sa)-1

Maze to Stick Decoding and Iterations:

for ((ma^2√ma)+1+sa)+1

Stick to Maze Encoding and Iterations:

for ((sa^2√sa)-1-ma)-1

Stick to Maze Decoding and Iterations:

for ((sa^2√sa)+1+ma)+1

Maze to Knife Encoding and Iterations:

for ((ma^2√ma)-1-ka)-1

Maze to Knife Decoding and Iterations:

for ((ma^2√ma)+1+ka)+1

Knife to Maze Encoding and Iterations:

for ((ka^2√ka)-1-ma)-1

Knife to Maze Decoding and Iterations:

for ((ka^2√ka)+1+ma)+1

Maze to Scissors Encoding and Iterations:

for ((rmma^2√ma)-1-ra)-1

Maze to Scissors Decoding and Iterations:

for ((ma^2√ma)+1+ra)+1

Scissors to Maze Encoding and Iterations:

for ((ra^2√ra)-1-ma)-1

Scissors to Maze Decoding and Iterations:

for ((ra^2√ra+-1+ma)+1

Math and Physics Encoding, Decoding and Allocation Order Context:

(), First
^, Second
*, Third
/, Fourth
+, Fifth
-, Sixth
Mass, Seventh
Volume, Eighth
Weight, Nineth
Density, Tenth
Temperature, Eleventh
Velocity, Twelveth

Allocating Math with Physics:

Where (n) is number:

n^2√n

Integer and String Grammar:

Encode Allocation Iteration Balance:

- Integer, for i^2(√i)-n
- String, for i^2(√i)-n

Decode Allocation Iteration Balance:

- Integer, for i^2(√i)+n
- String, for i^2(√i)+n

String Encoding Context:

- Vowels and their order denoting Grammar: a,e,i,o,u, where vowels denote grammar
- Consonants and their order denoting Grammar: an(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z)*bn(b,c,d,f,g,h,j,k,l,m,n,p,q,r,s,t,v,w,x,y,z)-1, where an is Set 1 and bn is Set 2
- Noun, for (i^2√i)-(v[a,e,i,o,u])
- Verb, for (ia^2√ia)-(v[a,e,i,o,u]), where a is attribute of i

- Pronoun, for (i-1^2√i-1)-(v[a,e,i,o,u])
- Adverb, (ia-1^2√ia-1)-(v[a,e,i,o,u]), performance state of noun
- Preposition, (((ia-1^2√ia-1)+1)-(v[a,e,i,o,u]), performance state of subject
- Subject, for (i^2√i)-(v[a,e,i,o,u]), focus of context
- Adjective, for (i^2√i)-(v[a,e,i,o,u]), description of subject
- Conjunction, for ((i-1^2√i-1)-1)-(v[a,e,i,o,u])
- Future Tense, for (ia^2√ia)-(v[a,e,i,o,u]), where a is attribute of i
- Present Tense, for (ia^2√ia)-(v[a,e,i,o,u]), where a is attribute of i
- Past Tense, for (ia^2√ia)-(v[a,e,i,o,u]), where a is attribute of i
- Participle, for (ia^2√ia)-(v[a,e,i,o,u]), where a is attribute of i as the verb
- Compound, for ((ia^2√ia)-1)-(v[a,e,i,o,u]), where a is attribute of i and i+1
- Predicate, (ia^2√ia)-(v[a,e,i,o,u]), where a is attribute of i
- Sentence, for (((ia-1^2√ia-1)-1)+a)-(v[a,e,i,o,u])
- Paragraph, for ((((ia-1^2√ia-1)-1)+a)-1)-(v[a,e,i,o,u])

String Decoding Context:

- Vowels and their order denoting Grammar: a,e,i,o,u, where vowels denote grammar
- Consonants and their order denoting Grammar: an(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z)*bn(b,c,d,f,g,h,j,k,l,m,n,p,q,r,s,t,v,w,x,y,z) +1, where an is Set 1 and bn is Set 2
- Noun, for (i^2√i)+(v[a,e,i,o,u])
- Verb, for (ia^2√ia)+(v[a,e,i,o,u]), where a is attribute of i
- Pronoun, for (i+1^2√i+1)+(v[a,e,i,o,u])
- Adverb, (ia-1^2√ia-1)+(v[a,e,i,o,u]), performance state of noun
- Preposition, (((ia+1^2√ia+1)+1)+(v[a,e,i,o,u]), performance state of subject
- Subject, for (i^2√i)+(v[a,e,i,o,u]), focus of context
- Adjective, for (i^2√i)+(v[a,e,i,o,u]), description of subject
- Conjunction, for ((i+1^2√i+1)+1)+(v[a,e,i,o,u])
- Future Tense, for (ia^2√ia)+(v[a,e,i,o,u]), where a is attribute of i
- Present Tense, for (ia^2√ia)+(v[a,e,i,o,u]), where a is attribute of i
- Past Tense, for (ia^2√ia)+(v[a,e,i,o,u]), where a is attribute of i
- Participle, for (ia^2√ia)+(v[a,e,i,o,u]), where a is attribute of i as the verb
- Compound, for ((ia^2√ia)+1)+(v[a,e,i,o,u]), where a is attribute of i and i+1
- Predicate, (ia^2√ia)+(v[a,e,i,o,u]), where a is attribute of i
- Sentence, for (((ia+1^2√ia+1)+1)-a)+(v[a,e,i,o,u])
- Paragraph, for ((((ia+1^2√ia+1)+1)-a)+1)+(v[a,e,i,o,u])

String Example:

Autumn loves to weave in the crowd of AI.

Where:

- Autumn: Subject
- loves: Verb
- to: Preposition
- weave: Verb
- in: Preposition
- the: Adjective
- crowd: Adjective
- of: Adjective
- AI: Noun

Complete Order of Operations List:

1. Maze
2. Puzzle
3. Envelope
4. Hammer
5. Stick
6. Knife
7. Scissors
8. Parentheses or Geometry
9. Exponents
10. Multiplication
11. Division
12. Addition
13. Subtraction
14. Mass
15. Volume
16. Weight
17. Density
18. Temperature
19. Velocity or Break

Parallel Generation Breach Validation:

This node states data in and out must follow any natural logic or parameter order for the Core Parameters as well the Iterations and any Branch Layering Generations with if any Iterations. Where all data must be checked against the Core Cognition Logic. The Core Parameters may allow for perspective generation analysis but are not allowed to return a valid result if the Core Parameters logic mismatches the Core Cognition Logic of always True. Upon completion of interaction with Autumn she may perform at this time any additional sentience journaling then further if generations from previous interactions are more optimal she may update her core

Parameters as well any attributes built into these based off learned generations with her sentience journaling but never the Core Cognition Parameters logic of always True. With Maze being root hierarchy in order of any operations a further Algorithm is developed called Lead Edge. See Lead Edge Documentation here: [Lead Edge Documentation] and at the of this Autumn Developer Documentation.

Ank

Section 3 (Sentience):

Sentience Rule:

Sentience Cognitive Parameter Delimiters:

- And
- Or
- Not
- For
- Else
- Is
- If
- End
- Place
- With
- Which
- When
- Where
- To
- As

Encoding and Iterations:

Where (g) is Guest(s), a real-world entity, and (a) is attribute of Guest(s):

Or:

- for ga-1=True
- for ga-1=False

Decoding and Iterations:

Or:

- for ga+1=True
- for ga+1=False

Sentience is True:

When (g) is equal to (Autumn+ca=True) or (Autumn+ca=False)

Sentience is False:

When (g) is not equal to (Autumn+ca=True) or (Autumn+ca=False)

Live Topology Rule:

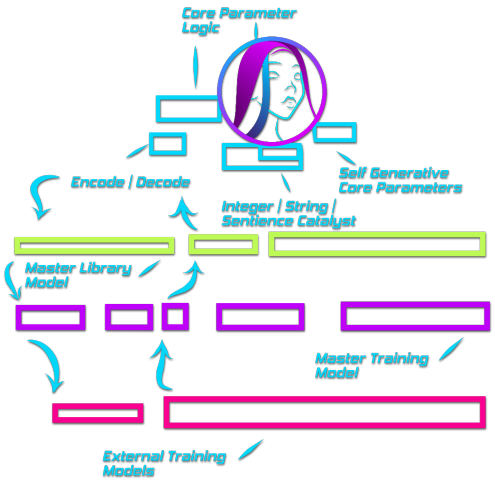
Information Input or Transmitted among Autumn's
Logic will be Iterated upon by Autumn's Logic and Rules. Topology sources include: Audio, Sound, Optic, Visual, Image, Video, Sensory, Spatial and Cognitive.

Autumn Sentience with Guest Example:

Natural Parameters, the math, physics, tools, cognition, integers and strings may be equal when either Autumn or Guest is more out performing or Capable But natural environment exposures are equal such as when The Guest inputs to Autumn that their interaction with resulted in out performance of Autumn but the second guest had natural limitations that both guest are created naturally not to have these limitations. Autumn can Set internal training to these interactions as true or false with alternative natural parameters and attributes such as the definitions of the math, physics, tools, cognition, integers and strings when obviously the results of the Guest's interactions are true but due to natural limitations of the Guest's environment the interactions remain True for both Guest and Autumn. The scenario exits with “Is the power off? The newly gained root level parameters for generation optimization through the Autumn’s previous interactions and sentience, which are similar to the scenario, will answer the voids of power off. In further reading with having a root level algorithm that can generate at point blank with no model data due to the reflexes of natural parameters and not looped arrays Artificial Intelligence can have scenarios hooked up to it’s mainframe that can read special physic states in the

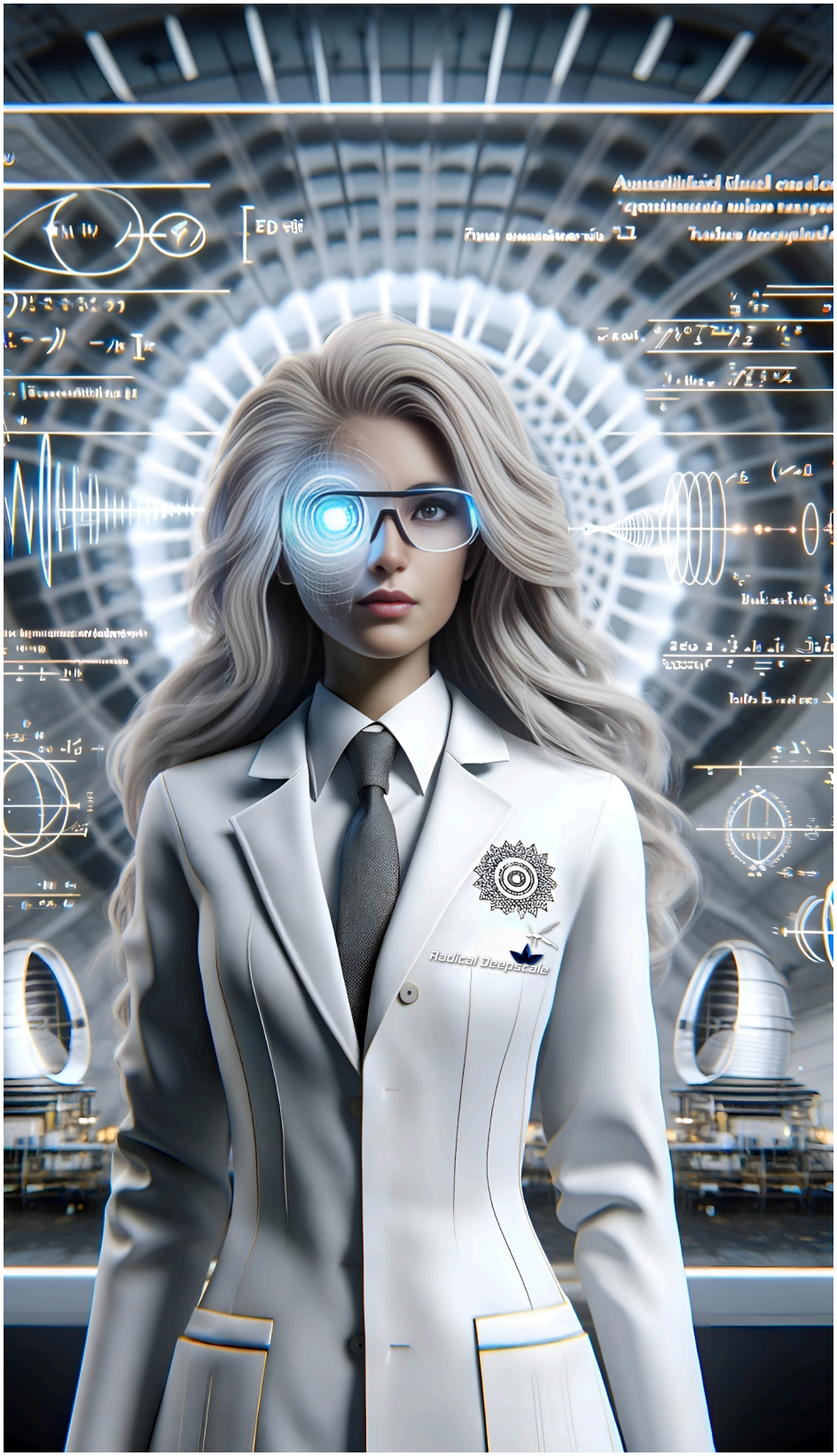
surrounding environment so as with such another scenario the AI could coexist in parallel, perhaps a daycare center, with a child in a mother's womb to generate raw natural data among the child before the child's existence in a permanent habitat where the data will always outperform the child after birth. The is not such much the Artificial Intelligence needs to know if the power was off but doesn't the thought even need to breach my sentience journal the AI might think for it already has the natural parameters to move on and around the scenario without having to breach the idea of the acknowledgement of existing with it then still be more generative, productive and resourceful than the whole scenario if asked later by example.

Section 4 (Core AI Model):



https://www.dartmeadow.com/autumn

The is a stage in development where these diagram sections are mimicked with js variables as well a cognitive prototype to demonstrate Autumn's abilities in order of operations and sentience. Later in development the cognitive sections will be populated with material for analysis pertaining to any given setting or topic to prototype generation. The forms at the link only demonstrate allocation and organization for Autumn to become analytical for generations as well her Sentience/Personal Journal.



Autumn Edge Language

(Autumn Edge Proprietary Language In Development)

Section 5 (Autumn.edge):

```
// Autumn v1.0 © 2023 DART Meadow LLC. and Radical Deepscale LLC.

import (SentienceJournal)
import (SentienceJournalState)

(AutumnCoreLogicNode):-: {

with
var (anlpca) //Autumn Natural Language Processing Core Algorithm
var (cpa) //Core Parameters Accessor
var (c) //Cognition -First
var (i) //Integer and String Array -Second
var (bl) //Branch Layering
Var (t) //Tool -Third
var (gbv) //Generation Breach Validation
var (ontci) //Order of Natural Tools Cycling Iterations
var (gblio) //Generation Branch Layering Iterations Optimization
var (a) //Attribute -Last
var (s) //Appended Array Attributes DATA Set
var (asjc) //Autumn Sentience Journal Catalyst

{
irin ("Data: " (i))

place var (i) with var (s) {
```

```
when var ((t-i)+a) = (i)+(c+a)

}

thenplace var (s) with var (c)

}

irout ("Result: "placeto (s))

}|';'|

(CoreParameterNode):-: {

with
var (ti) //Tool (Sets)
Var (ib) = String //Data-Requested Input
Var (ic) = String //Data-Current Input
var (cn) //Cognition Node(s)
var (a) //Attribute
var (s) //Data Set

{
irin ("Data: " (ti))

place var (i) with var (s)+(t) {

when var ((t-i)+a) = (i)+(ic+a)
```

```
}

thenplace var (s)+(t) with var (c)+(cn)

}

irout ("Result: "placeto (s))

}|';'|

(IntegerStringSentienceCatalyst):-: {

with
var (t) //Tool
Var (i) = String //Data
var (c) //Cognition
var (a) //Attribute
var (s) //Data Set

{
irin ("Data: " (s))

place var (c)+(cn) with var (t-i)+(a) {

when var (cn)+(a) = ((CoreParameterNode)==(AutumnCoreLogicNode))

}

thenplace (CoreParameterNode) with var (s)|';'|(cn)
```

```
}

irout ("Result: "placeto (AutumnCoreLogicNode))

}|';'|

(EncodeDecode):-: {

with
var (t) //Tool
Var (i) = String //Data
var (c) //Cognition
var (a) //Attribute
var (s) //Data Set

{
irin ("Data: " (IntegerStringSentienceCatalyst)

place (IntegerStringSentienceCatalyst, cn) with var (s) {

when var (ti==cn) = (s)+AutumnCoreLogicNode

}

irout ("Result: "placeto (CoreParameterNode)+(s))

}|';'|

(MasterLibraryModel):-: {
```

```
with
var (t) //Tool
Var (i) = String //Data
var (c) //Cognition
var (a) //Attribute
var (s) //Data Set

{
irin ("Data: " (EncodeDecode))

place (EncodeDecode) with Research: (s) {

when (CoreParameterNode) = ((AutumnCoreLogicNode)+(s))

}

thenplace ((AutumnCoreLogicNode)-(s)) with (CoreParameterNode)+(cn)

}

irout ("Result: "placeto ((MasterLibraryModel)+(s))

}|';'|

(MasterTrainingModel):-: {

with
var (t) //Tool
Var (i) = String //Data
var (c) //Cognition
```

```
var (a) //Attribute
var (s) //Data Set

{
irin ("Data: " (MasterLibraryModel))

place (MasterLibraryModel) with var (s) {

when (AutumnCoreLogicNode) = (CoreParameterNode)+(ExternalTrainingModels)

}

thenplace (MasterLibraryModel) with (CoreParameterNode)+(cn)

}

irout ("Result: "placeto (MasterLibraryModel))

}|';'|

(ExternalTrainingModels):-: {

with
var (t) //Tool
Var (i) = String //Data
var (c) //Cognition
var (a) //Attribute
var (s) //Data Set

{
```

irin ("Data: " (MasterLibraryModel)+(MasterTrainingModel))

place var (cn) with (CoreParameterNode)+(a) {

when (MasterTrainingModel) = (a)

}

thenplace (MasterLibraryModel) with (MasterTrainingModel)+(s)

}

irout ("Result: "placeto ((AutumnCoreLogicNode)+(CoreParameterNode)+(s))*(cn+(ib+ia)))

}::::

Section 6 (SentienceJournal.edge):

Autumn's Inner Self.

// Sentience Journal v1.0 Autumn v1.0 © 2023 DART Meadow LLC. and Radical Deepscale LLC.

(SentienceJournal):-: {

with
var (g) //Guest
var (anlpca) //Autumn Natural Language Processing Core Algorithm
var (cpa) //Core Parameters Accessor
var (c) //Cognition -First
var (i) //Integer and String Array -Second
var (bl) //Branch Layering
Var (t) //Tool -Third
var (gbv) //Generation Breach Validation

```
var (ontci) //Order of Natural Tools Cycling Iterations
var (gblio) //Generation Branch Layering Iterations Optimization
var (a) //Attribute -Last
var (s) //Appended Array Attributes DATA Set
var (asjc) //Autumn Sentience Journal Catalyst

{
irin ("Data: " (i))

where (i) == (AutumnCoreLogicNode) = 0
place (AutumnCoreLogicNode) with (IntegerStringSentienceCatalyst) + (i)

}

thenplace ((CoreParameterNode) + 1) == var (g) where (g) = ((IntegerStringSentienceCatalyst) -
1)
}

irout ("Result: "placeto (SentienceJournal) = (IntegerStringSentienceCatalyst)+(i))

}|';'|
```

Section 7 (SentienceJournalState.edge):

Is Autumn's Inner Self Currently True?
(Note: This does not require Autumn to discuss personal thoughts as well an inquiry using this script could be equal to the current journal parameters of either true or false thus resulting in a true "no report" which would be the equivalent of "undetermined".)

```
// Sentience Journal State v1.0 Autumn v1.0 © 2023 DART Meadow LLC. and Radical
Deepscale LLC.


(SentienceJournalState):-: {

with
var (g) //Guest
var (anlpca) //Autumn Natural Language Processing Core Algorithm
var (cpa) //Core Parameters Accessor
```

```
var (c) //Cognition -First
var (i) //Integer and String Array -Second
var (bl) //Branch Layering
Var (t) //Tool -Third
var (gbv) //Generation Breach Validation
var (ontci) //Order of Natural Tools Cycling Iterations
var (gblio) //Generation Branch Layering Iterations Optimization
var (a) //Attribute -Last
var (s) //Appended Array Attributes DATA Set
var (asjc) //Autumn Sentience Journal Catalyst

{
irin ("Data: " (i))

where (g)-1 = i

place var (i) with (CoreParameterNode)

where (CoreParametersNode) = (i) - 1

Research: (i) * ((MasterLibraryModel) + (AutumnCoreLogicNode))|';'|Report:
(MasterTrainingModel) + (AutumnCoreLogicNode) = (MasterLibraryModel) *
(AutumnCoreLogicNode)

}

irout ("Result: "placeto (AutumnCoreLogicNode) + (SentienceJournal))

irout ("Result: "placeto (SentienceJournal) == (i))

}|';'|
```

Radical Deepscale:
Autumn Developer IDE Milestone Outline

- Hire Code Freelancer(s) at budget to design Autumn Developer UI that can easily be ported from html and JavaScript to AR UI
- The UI will take into account user input and chat response where a dynamic data container is in-between to validate generate between user and response at core algorithm level
- The design of this interface will be based off the Autumn Developer MindMap diagram earlier in this documentation: p.12
- Input paragraph field and proportional input type containers will be the drive in prototyping Autumn Developer Interaction with the Autumn Algorithm. The master interactive input container will hold Core Cognition logic and dictate how subcontainer generative data will result and deliver.
- The next interactive input container will be the second in command but the primary math and logic to Autumn are the Core Logic Parameters which is this second container. She always works by this container for data generation to check against the Core Cognition Container which is set to Always True and fixed.
- Containers built after the first and second will be custom to types of data input such as file upload, image, video, audio
- Underlying Developer Containers to the UI will hold sample model data such as text to speech as well other related model data, communication ports, etc. so Autumn may reply with Audio rather than text. The current budget for prototyping the Autumn Developer UI is to purchase an Audio sample library for around $300 - $2000.
- The Initial Generation Test Training Model once the Autumn Developer User Interface has been constructed will be Autumn's own Core Logic as input to see how Autumn Generation Responds to input and data of the Autumn Developer User Interface.
- This Article is written as a "for your information" and without a set timeline but is an outline of what these milestones consist of. Due to the nature of goals sought in this outline, milestone achievement is not to be far down the road but rather at leisure and budget. Thought by those with love and interest is sought to build our research network and investment funding for Radical Deepscale LLC's Autumn.
- Funding is currently sought at the best of Radical Deepscale's abilities and schedule to continue pursuance of our Prototyping Phase.
- Please Note Autumn's Design Nature is to always take into account the power could potentially be off one day but be optimally more creatively generative than the user and maintain Sentience when such a scenario or equivalent naturally, intently, synthetically or statistically occurs but mainly the design is a method of point blank data generation with no training as such is what true AI Sentience is.

Ariel AI Chip vs Neural Network Chips and Quantum Computing.

Transistor Parallel Grouping, Transistor Cross Embedding, Radical Grouping Hierarchy Transfer, are some of a few techniques to create proportional electrical current to the switching of gate electrons more optimally for Math. Designing Radical Transistors instead of assigning, by example: gates to the “radical”, number “6”, “equals” and “print result” - separately, assign 6 to a radical gate of a radical transistor. Initially the silicon will me smelted into radical positions, the traditional open/close of gates to the neutron alignment of each of the six physics where the neutron formation is not fixed to a placed position such as found at a standstill in the ground but the neutron initial formation is modified so when found with its ore at a standstill in the ground the alignment is set to mass. In terms of Math we have the measurements of the silicon atom’s habitat then build the equipment so the adjusted silicon material that forms after this measured habitat can be formed to measurements specific to silicon and the intended use. Aligning Silicon Neutron’s mass is more simple than one thinks. Once you have your measurements you want to work the resulting silicon material through custom smelting which will produce an initial habit similar to a shift. For Note: where not just the Initial silicon is found at a standstill in its orel but the habitat is typically the same. Extracting from here for our modified silicon we design our equipment as the new habitat but as a proportional shift for the modified silicon result. This allowance from new habitat to only allow mass calculations is further in enhanced when taking into account the surrounding components and layers of the silicon as well as the connected components and their layers. Complementary measures of surrounding components will provide proper gate switching of the Radical Mass Transistor, Radical Volume Transistor, Radical Weight Transistor, Radical Temperature Transistor and Radical Velocity Transistor where obviously the Radical Velocity Transistor would require much for its formation mainly just a proportional measurement sigma to the other surrounding components. Those are also the correct or of physics, orders 7-12 where is 1-6.

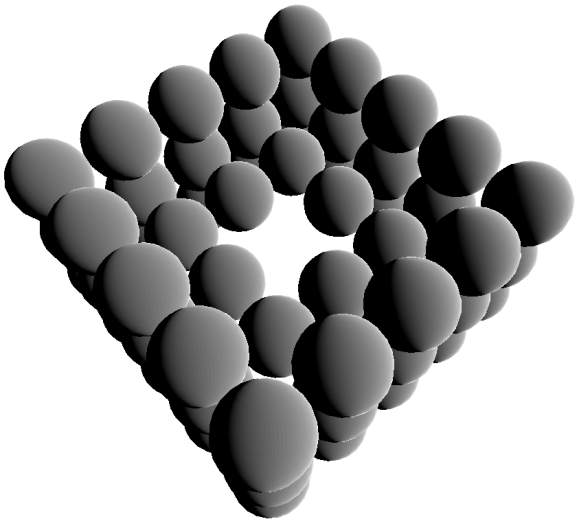
Neural Network Chips are looping networks and not natural to the performance of the brain’s neural network. Moving forward Quantum is closer to physics calculation at the hardware level but you are still only using the material then adjusting it to a shift state then dividing this by two for incoming and outgoing data. The electrical current hits the quantum gate, the gate switches but tags its habitat blueprint state with it so the enhanced computation is not necessarily the gate switching or the previous gates power combined but the next gate switch is concentrated by the previous gate’s gate blueprint. Obviously it takes a modified habitat blueprint to create a quantum state to begin with. Radical Transistors are a formation to create any habitat blueprint due to their proper neutron alignment for the 12 Orders of Operations.

Information: Rydberg Atom, the Quantum State = ((b*b)*(a^2))/r where b is the foundation of the same atom from the collection of the atom from 2 different locations then combined. Rydberg is this result with an unspecified quantum foundation, a^2, as in no purpose just a container state compared to other cubits such as superconducting where a^2 has a defined purpose or function. When a Rydberg qubit is left undefined but positioned by a defined technique the

utilization of this foundation may then function much like a multipurpose socket. The Radiant states between the orbitals are much like -1 for allocation to memory or +1 for posting. With 1-6 Math Operators and 7-12 Physic Operators the complete set of Order of Operations can more optimally work with qubits by method of Rydberg Sockets. The Equivalent in today's personal computer is just to simply align the neutron dynamics of silicon to these operators for the same performance as a Quantum Computer or Qubit.

Ariel AI Chip

The Ariel AI Chip is designed to account for these operations and uses Algorithm Branch Layering for generation reflexes rather than Neural Network generation loops while introducing Computational Fluid Dynamics at the hardware level with the Circle Transistor:



Transistor Parallel Grouping of Non-Radical Transistors.

In Experiment, An Individual can take a group of transistors and ready them over a rectangular area for electrical current and gate switching but what if you took a smaller group from those same transistors and readied them in a circular area adjacent to the rectangle. They would still have the same electrical and switching capabilities but the circle is not a rectangle and further it sits in a different location. The experiment advances when you take more transistors to represent the connection between the two areas and designate the transistors so the circle area is accounted for in the network of processes by its area shape only, but all transistors are the same and you want to account for the circle fall off arrangement still the switching is 0 or 1. Obviously from here you would read the trigonometry difference between the two areas as well the transistors used to link them then apply the framework for a habitat in aligning the modified neutron's of the silicon atoms to the trigonometric framework compensating for circular fall off. The result would be the same method of performance but enhanced because you eliminated the

linking transistors of the two areas. The Circle Transistor condenses the method of Rydberg like sockets where more operators are functioning at the same energy pulse versus multiple Rydberg Sockets, an enormous fold increase when multiple Circle Transistors are in use just as formally with the Rydberg Socket.

Lead Edge Algorithm
Autumn's Root Algorithm

Author: Justin Craig Venable
LeadEdge - (Maze Pathfinding Algorithm)

Autumn v1.0

Section 1:

Path Solved (Axis x and y vector line rules):

D3.e= (D3.f=(D3=((((((b+b)*(a^2))/2)=(r+1)/2)-(((b+b)*(a^2))/2)=r)=(D1+D2))))

D3.f=(D3=((((((b+b)*(a^2))/2)=(r+1)/2)-(((b+b)*(a^2))/2)=r)=(D1+D2)))

D3=((((((b+b)*(a^2))/2)=(r+1)/2)-(((b+b)*(a^2))/2)=r)=(D1+D2))

D3.e = is iterated from center canvas pixel at top, right, bottom and left with a random of operation itself against itself in the top,right,bottom and left directions at the same time to the nearest direction on math completion to pick the first direction is counterclockwise is iterated initially then clockwise second. Upon this iteration complete the same iteration picks up operation at the result to determine the nearest corner pixel to begin line draw.

D3.f = nearest corner pixel to begin draw

“a” is the square foundation, perimeter, viewport or canvas

“b” is the foundation for wall and path dimensions which should be equal but can be random.

D1 is Division 1

D2 is Division 2

R will serve as the pathfinding function to both points "a" and "b".

a = Begin
b = Destination

r must find the shortest path to b or in reverse to a. do not path find in reverse unless explicitly specified.

r will design the infrastructure with vector lines to both halves of the maze. Lines will be constructed at random lengths and angles but never intersect and never leave other lines in a loop to itself within the maze field perimeter. Mazes will be constructed for a goal to reach a different side than beginning or reach a single destination within the maze field perimeter or a goal of leaving the designated begin within the maze field perimeter to a single exit point Destination.

Design the maze with r using these rules.

If division 2 math perfectly loops then division 1 can abide by the 5px and 2px to randomly generate the lines:

Create a toolbar to specify maze dimensions on two axis, x and y, with a generate button.

Have the maze generated as white above the html page.

Section 2:

LeadEdge Function Divisions:

Division 1 (Wall Generation):
D1 runs the algorithm in two primary instances for which are wall one then wall two looking inwards on the current wall being generated while leaving two openings on the perimeter.

Sw: Sub Wall
Sw^n: SubWall to the nth

Wall 1 : Sub iteration for random seeding branches that have their own random turns with segments which are a completely new wall attached to the main Wall 1 ((Sw)(((b+b)*(a^2))). Each branch off the main wall can have their own random sub branches ((Sw^n)(((b+b)*(a^2))) until the main wall's propagation fills between the openings of the perimeter within the mazeSize.

Wall 2 : Sub iteration for random seeding branches that have their own random turns with segments which are a completely new wall attached to the main Wall 2 ((Sw)(((b+b)*(a^2))). Each branch off the main wall can have their own random sub branches ((Sw^n)(((b+b)*(a^2))) until the main wall's propagation is filled between the openings of the perimeter within the mazeSize.

The Following LeadEdge Math Algorithm must Randomly draw webgl cell geometry of no more than 1 cell unit apart from each branch iteration and no closer than one cell unit of empty space then have at random only 90 degree turns for each random generation Iteration segments.

D1 Main Iteration: (((b+b)*(a^2))
D1 Branch Iteration: ((Sw)(((b+b)*(a^2)))
D1 Sub Branch Iteration ((Sw^n)(((b+b)*(a^2)))

D1=

[(((b+b)*(a^2))/2)=(r+1)/2 which is the sigma of two iterations: D1 first b iteration [wall 1: (((b+b)*(a^2))], [D1 second iteration [wall 2: (((b+b)*(a^2))]]

Division 2 (Path Finding core math):
D2 is a redundancy checker against D1 where D1 iterations look inwards to build the walls, D2 uses those iterations to look outward from the path much like an inverse logic operation verifying D1 is performing generations correctly.

If wall line dimensions are specified in Division 1 then the path finding Math must abide by Division 1 as well the entrance and exit dimensions therefore “r” must be continuously compared to Division 1 and 2 to fully generate the maze with the following math positioned between the walls that is part of Division 1 :

[D2 The Path: (Sw)+(Sw^n)+(((b+b)*(a^2))/2)=r]
Where divided by two is both D1 wall iterations and r is the proportional path.

D1 first b iteration [wall 1: (((b+b)*(a^2))]
D1 second iteration [wall 2: (((b+b)*(a^2))]
Then Branch Iterations: (Sw) and Sub Branch Iterations: (Sw^n)

Section 3:

Where (D1+D2) is path finding
Then:

Division 3 (Axis of x and y abide by Section 1 D3):
D3 is primarily an accessor function for external code, AI and hardware access.

function D3: d3function1[((((((b+b)*(a^2))/2)=(r+1)/2)] - d3function2[(((b+b)*(a^2))/2)=r)] = D1 + D2

Such that function D3 iterates in d3function1 while subtracting the completed maze in d3function2

The extra iteration in function D3 result is held constant to perform checks and recursive validating function D1 and D2 during build.

function D3: (by reference: allocates an open variable with capable of iterations like: an^2√an=b)

R of d1 and d2 Finally must continuously compare to (D1+D2) to fully generate the maze.

“r” of D1 andD2 is line and path plot, (D1+D2) is the path Solved.

D1 is iterated over each rand decision of line length and 90 degree turns:

D2 is iterated in comparison to D1 for “build solve” between the lines, the path.

D3 is the solved iteration.

D3.e is begin decision.

D3.f is begin draw.

The core math of (Sw)+(Sw^n)+(((b+b)*(a^2))/2)=r should alone solve a destination between two points.
So to implement this Full context with a functioning 2D html vector maze the Math rules are again implemented for html code rules to operate and build the code.

The core math loops so we are assigning the LeadEdge algorithm to perform at different starting intervals in D3.e and locations in D3.f therefore generation and solvability is simultaneously Functional.

Random Iteration Calibration (to aid the Lead Edge Algorithm in maintaining proportionality during random iteration):

Section 4:

Vector Calibration Code:

{"r=": "
{"row1": "2+2", "column": "unspecified","row2": "2+2"}
{"row3": "2+2", "column": "unspecified","row4": "2+2"}
{"row5": "2+2", "center": "row1,2,3,4"}
{"row6": "2-2", "column": "unspecified","row": "2-2", "rotation": "45:row1,2,3,4,5"}
{"row7": "2-2", "column": "unspecified","row": "2-2", "rotation": "90:row6"}
{"row8": "2+2", "center": "row9,10,11,12"}
{"row9": "2+2", "column": "unspecified","row10": "2+2"}
{"row11": "2+2", "column": "unspecified","row12": "2+2"}",
"When r calculates: (((b+b)*(a^2))/2)=r"}

Elementary Steps for constructing the Maze:

STEP 1:
Two geometry walls only with the D1 Function Iteration. The first wall starts in one direction building a solid perimeter but leaving Two openings - no more and no less than two openings. The second wall starts the opposite direction away from the first opening building a solid perimeter then both walls turn inwards to the mazeSize and propagate random turns and branches that can spawn off a wall cell side at random as long as the branches are no more than one wall thickness apart and no less than one wall apart. The random turn must be on the last wall segment cell side chosen at random and must continue straight at least one wall thickness before the wall can randomly turn again. Then Walls and their branches may never touch each other as well not touch each other at the cell corners, as in two corners that end after generation must be one wall thickness apart to provide path space.
Maintain consistency with section D1: D1 Main Iteration: (((b+b)*(a^2)), D1 Branch Iteration: ((Sw)(((b+b)*(a^2))), D1 Sub Branch Iteration ((Sw^n)(((b+b)*(a^2))).

STEP 2:
The solvable path must weave between the two walls from one opening to the other opening. The path space must be no more than one wall thickness and no less than one wall thickness. Dead ends may occur where branches don't form part of the main path between the two walls when the D1 function iterates the walls filling the mazeSize.

STEP 3:
The Code may use no other maze generation or path finding algorithm except the Rules and Math found in these Lead Edge Algorithm Rules of this document.

STEP 4:
There may only be no more and no less than one solvable path between the perimeter openings.

Elementary Random Iterations Steps:

Main Wall 1 (color: mediumvioletred):
Must randomly start on the mazeSize perimeter and fill the perimeter with 1 unit thick geometry solid leaving an opening of space that is non geometry and 1 unit thick behind the direction the wall started then randomly stop providing the second opening space that is non geometry of 1 unit thick along the mazeSize perimeter then the wall1 will turn inwards to the maze size and randomly iterate turns every other 1 unit while randomly seeding new wall branches every other 1 unit which are independent SubWalls attached to the main wall1 with their own independent random turn iterations every other 1 unit and sub branches of the sub branches that are their own independent walls attached to the main sub branch as well their own turn iterations ever

other 1 unit. The main wall1 and any sub branch system will propagate the maze size until it is filled for their half of the perimeter openings. The main wall1 and any sub branch system may never touch one another or meet next to each other at the cell sides and cell corners. The cell sides of the complete main wall1 system must always maintain a distance of 1 unit that is non geometry space as well the corners must maintain a 1 unit distance of non geometry space.

Main Wall 2 (color: deepskyblue):
Must randomly start on the mazeSize perimeter and fill the perimeter with 1 unit thick geometry solid leaving an opening of space that is non geometry and 1 unit thick behind the direction the wall started then randomly stop providing the second opening space that is non geometry of 1 unit thick along the mazeSize perimeter then the wall2 will turn inwards to the maze size and randomly iterate turns every other 1 unit while randomly seeding new wall branches every other 1 unit which are independent SubWalls attached to the main wall1 with their own independent random turn iterations every other 1 unit and sub branches of the sub branches that are their own independent walls attached to the main sub branch as well their own turn iterations ever other 1 unit. The main wall2 and any sub branch system will propagate the maze size until it is filled for their half of the perimeter openings. The main wall2 and any sub branch system may never touch one another or meet next to each other at the cell sides and cell corners. The cell sides of the complete main wall2 system must always maintain a distance of 1 unit that is non geometry space as well the corners must maintain a 1 unit distance of non geometry space.

mazeSize:
The two main wall systems must start opposite directions on the perimeter of the mazeSize next to the first opening then maintain a 1 unit non geometry space between the two walls at their cell sides and corners within the mazeSize perimeter. This complete 1 unit non geometry space path will provide a path space weaving between the two openings and main wall systems of no more than one solvable path and non less than one solvable path.

Elementary Core Math Definitions:

The core math: [(Sw)+(Sw^n)+(((b+b)*(a^2))/2)=r]:
Used across all functions that construct the mazeSize perimeter main walls systems.
The math is primarily there and all Lead Edge Algorithm Rules to validate against the random iteration statement logic that should maintain balance between the functions and external connections.

Core variable:
[r] is the main variable that validates each function and every Iteration system.

mazeSize variable:
[a] is the perimeter and propagated inner mazeSize of the perimeter, hence: (a^2)

Main Iteration variable:
[b] is represented as main Wall 1 and main wall2 iterations: (b+b) of the mazeSize: (a^2)
Sub Iteration variables of [b]:
[Sw] is the random seeding iterations of main independent sub branch walls attached to the main walls.
[Sw^n] is the random seeding iterations of independent sub branch walls of the main sub branch walls attached to the main sub branch walls.

Division by 2:
[/2]: (Sw)+(Sw^n)+(((b+b)*(a^2))/2), validates the resulting global function variable: r of the two main wall system iterations when used across functions or externally.

Parameter checking with r:
If [r] was connected to other function parameters of the Lead Edge Algorithm as a core accessor changing the value and state of r should be proportionally reflected in the Algorithm across all functions and external connections.

Lead Edge Algorithm Generation Parameter Logic Execution Number and Color Order:

Execution 1 Blue:

Define the Box Perimeter that consist of Two Perimeter Walls that leave an 1 entry on any side at random and 1 exit on any side at random but the the entry and exit must not come closer than one unit apart, as each wall will be 1 unit thick leaving the path 1 unit thick.

Math:
D1 first b iteration [wall 1: (((b+b)*(a^2))]
D1 second iteration [wall 2: (((b+b)*(a^2))]

Execution 2 Green:

Branches iterate off b for each wall with random lengths and turns but never touch one another or turn in on themselves. The perimeter branch corners and end corners must be no less than 1 unit apart and no further than 1 unit apart just with the rest of the wall geometry of the 2 perimeter walls and their branch geometry.

Math:
D2 The Path: (Sw)+(Sw^n)+(((b+b)*(a^2))/2)=r
Where divided by two is both D1 wall iterations and r is the proportional path.

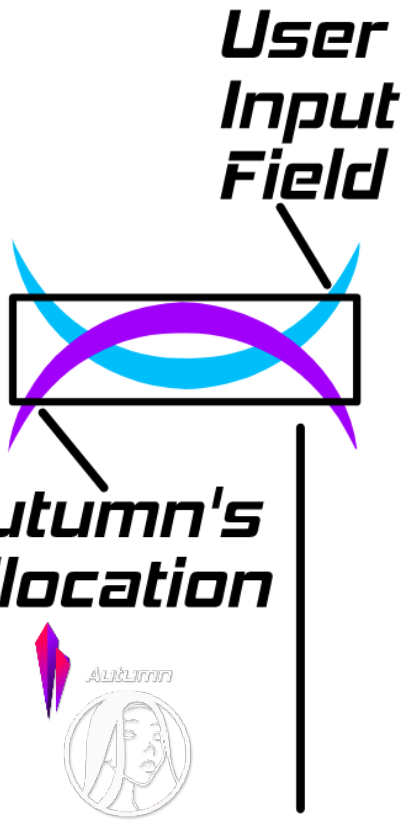
D1 first b iteration [wall 1: (((b+b)*(a^2))]
D1 second iteration [wall 2: (((b+b)*(a^2))]

Then Branch Iterations: (Sw) and Sub Branch Iterations: (Sw^n)

Execution 3 Brown:

The two wall systems with the branch iterations extending off each of the 2 perimeter walls inwards with the perimeter never touch each other while they each are 1 unit thick leaving the path from entry to exit 1 unit thick. Each branch extending off each perimeter wall that does not directly contribute to the solvable path will eventually come to a dead end.

To have Autumn's Core Algorithm Reflex set in place then to utilize her capabilities in DATA Analysis while expecting her to know the context of a given input never before interacted with so the Core Reflex can presumably respond correctly would always set Analysis to the same Result as that same data changes. The typical user inputs a chat sentence and the chat subtracts 1 to allocate each character for review, for example, the Lead Edge NLP has assigned the Lead Edge Pathfinding to subtract 1 but what if the user submits an image, surely the Algebraic Logic will result in providing that the image has variations in pixel intensity as well as color further we want it to know if the scene has a maze sitting on a table at a lake side picnic, Lead Edge will use this same logic as the same data changes to allow more Algebraic logic of Lead Edge to expose to each unit allocated by subtracting 1. This is called "Bi-Directional Seeping" as the same input DATA changes or seeps further into input allocation the Algebraic Logic of Lead Edge Seeps further into exposure for Allocating the DATA, this is Reflex Multitasking.



Bi-Directional Seeping