**SELENA**

An AI System

Revision 3

9/24/2022

By Walter Gress

Intellectual Property of Twilight Raven Games LLC

**Legal Disclaimer**

This research is proprietary intellectual property of Twilight Raven Games LLC. It’s implementation and redistribution is licensed under the GPL (GNU Public License)

Available for review at <https://www.gnu.org/licenses/gpl-3.0.en.html>

# Table of Contents

Introduction to Selena, an Integrative Data Process for an Autonomous Machine 5

System Goals and Structural Requirements 10

General Requirements for an intelligent, autonomous, machine Error! Bookmark not defined.

Introduction to Integrative Data Process for an Autonomous Machine Error! Bookmark not defined.

Requirements for Selena, a conscious computing system Error! Bookmark not defined.

Requirements for a purely autonomous machine 16

Additional Requirements for Selena specifically, as an autonomous machine 16

Expected requirements for a Free-Willed Machine 17

Expected requirements for Self-Instruction 17

Expected data structure components for Neuron Decision Making Units 19

Possible Use Cases for Neuron and Neurotransmitter Interaction Error! Bookmark not defined.

Expected requirements for a Natural Computing Language 19

Expected requirements for a Search Algorithm 20

Expected requirements for a Sort Algorithm 20

Expected Problems that may be encountered during development Error! Bookmark not defined.

Selena’s Three Operating Functions Error! Bookmark not defined.

Structural Error! Bookmark not defined.

Neuron clusters or “cores” 25

Language Error! Bookmark not defined.

M-Network Structure 27

Neurotransmitters Error! Bookmark not defined.

Notes on Machine’s Context Given Grammar and Context 33

Behavior and State vs Structure 37

Thought versus Memory 37

Mathematical Definition of an M-Network 38

Artificial neurons as processors, instructions, and node level decision making processes. 39

Actualization 43

Testing and Experiments 44

Input and Output on semantic and syntactical levels 48

Neuron Node Path Data Structure 54

Input and Output on semantic and syntactical levels 57

Neuron, as processors instructions, and the node level decision making process 64

Introduction 67

The Tautological Proof 67

Cross-Infection 67

Communication 68

Outcome 68

Testing the Primordial Ooze 70

# 

# Introduction to Revision 3 of SELENA

In the best interest of this paper and its contents I have decided to revise it for a third time and release it once again, now with its entirety established.

SELENA was originally devised over a decade ago while I was studying Artificial Intelligence at Drexel University, with the inspiration of creating a Map-Reduce algorithm that could be used over a Neural Network such as used in Deep Learning. The original Meta-Graph or “M-Graph” was modeled after a Semantic Network with nodes that established reinforcement and operations to implement logical inference of the connecting branches. I read today upon this day of document editing in the latest “Communications of the ACM” for October 2022 that inference is now being investigated as being implemented in the context of Neural Networks. I pondered this with the original release of the SELENA revision 1 documents. The original requirements were written separately, established before designing its algorithmic implementation. The disparity and incompleteness of this paper caused it to never be successful and I have come back to take the ideas I have learned to revise it in preparation for the release of a second paper, denoted SELENA 2, which builds upon this paper in a groundbreaking way.

The goal of the requirements detailed hereafter are to outline what is called a completely *autonomous* machine. In that I mean that the machine would work autonomously like a human being in that it would take inputs on its own and learn from them, without the drudgery of being manually tweaked like modern day Artificial Neural Networks., I admit that this was a lofty goal and it has taken me a significant amount of time to find myself able to return to this document and revise it for the following purpose. I believe with the current advances now in Artificial Intelligence, specifically with the implementations of the MapReduce algorithm this paper is more important than ever before. I hope to unify this paper with several ideas.

The MapReduce algorithm operates over key-value pairs. For its purposes it is quite impressive. However, the human brain, the most impressive organ of intelligence on Earth, has been modeled in AI as a network for the specific reason that the brain uses a path mechanism with forking/branching and abstraction to reinforce and perform its mapping of signals and respective input/output from various tissues and organs. In robotics this I/O is termed sensors and effectors.

We require a consistent set of algorithms to not only map these relationships as in a static Neural Network (with static being use in the regard to the fact that neural network pathways are defined and do not naturally transition but are only reinforced or punishment (the psychological antonym of reinforcement), but in a way in which we can introduce sorting and reduction. The ideas outlined in this paper are meant to do just that. The process has been simplified down to four essential functions. **Sort, Integrate, Search, Disintegrate.** These four functions when properly implemented, will allow an indexed Meta-Graph to take in new information, properly search for new information, and properly forget old information that iws no longer being reinforced regularly.

Semantic Networks, ANNs, MapReduce are all good starts. Now their ideas must be combined into a laid out, implementable, algorithmic system. That is SELENA 1.

# Introduction to Selena, an Integrative Data Process for an Autonomous Machine (from Revision 1, edited)

**A Machine’s Potential for Intelligence versus a Human’s Potential**

Machines, in general, have a *potential* for intelligence that is higher than the human brain’s potential. Given the task of a simple operation like counting by one for a sufficient amount of time, the machine will always produce a higher number of countable numbers than a man. In this case we are measuring a machine’s potential intelligence by the speed of its arithmetic operations versus that of a human brain’s. This potential for arithmetic computation speed and accuracy has been recognized for decades, maybe even since the inception of computational machines. The real limit has been the capacity of memory, which has been rapidly, if not exponentially, with advances of storage technology.

As prior mentioned, presently, machines have a lower *capacity* for intelligence due to the limited amount of storage space they have in comparison to man, whose storage space appeared to be in the past to be almost infinite, although recent research has shown that the human brain has its own limitations. I say near infinite because throughout the lifetime of man there is no specific limitation that can be observed on the amount of new knowledge he or she can collect, that there has only been a primary problem with recall and recognition. In the matter of machines, this amount, has been clearly defined and has clear limitations upon its scope.

What follows is a series of talking points on what differentiates an instructed machine, traditionally called a computer, and a fully autonomous machine, typically referred to as a computer with Artificial Intelligence Software (AI). When we talk of an autonomous machine our primary objective is the development of a machine which can instruct itself, that is, a self-instructing or fully autonomous machine. The ultimate goal of this paper is to present an algorithm based on an MapReduce, and a data structure called a Meta-Graph, that will be described in detail, that this algorithm can operate over to create a machine that can effectively learn with little training by self-instruction through the mapping received through its respective sensors and effectors. Sensors and Effectors int eh case of this research project are logical and not physical like in modern day Robotics, with hope that physical sensors and effectors can be implemented based off of these designs.

To be autonomous, a machine must be able to generate *useful* instructions for itself. The machine must be able to, given a task, devise a solution and instruct itself on how to carry out that solution. This is what I mean by *useful*. Another test of intelligence is: if given a series of instructions, can the one instructed, close to accurately, predict the results of its own instructions before executing them. This boils down to an advanced pattern recognition operation where a pattern is established by a set of instructions and that pattern is then extended, forming the prediction. The ability to predict the outcome of a series of events is necessary because it is a necessary step in planning, part of Executive Functioning.

**Definition: Executive Functioning**

( https://developingchild.harvard.edu/science/key-concepts/executive-function/)

**Executive function and self-regulation** skills are the mental processes that enable us to plan, focus attention, remember instructions, and juggle multiple tasks successfully. Just as an air traffic control system at a busy airport safely manages the arrivals and departures of many aircraft on multiple runways, the brain needs this skill set to filter distractions, prioritize tasks, set and achieve goals, and control impulses.

In order to accurately plan, generate its own instructions and understand instructions the system shall require a language system of some sort. Context-Free Grammars have been used in the past to instruct computer systems but there is no easy to way to instruct a Neural Network using a Context-Free Grammar. Natural Language, what I call a Context-Given Grammar, are the most common sensor input used for training the human brain and Natural Language Processing is a hot topic in AI. There are those researchers who have believed they have created sentient chatbots just by training them to understand and regurgitate natural language. This is obviously a prominent part of research and an important factor to consider in our proposed system. Natural Language chatbots have been investigated by Google using MapReduce as a base algorithm, training over corpuses, and using Neural Networks dedicated to this purpose. Markers are used to maintain *context* within the Neural Networks and to limit endless depth-first searching.

When it comes to language there is a disagreement between those who study sciences of the mind. There are some researchers that are of the opinion that we produce *thoughts* and from them produce words that form our *languages*. There are others of the opinion that *thought*, and *language* are synonymous. In the latter case language plays an important part in thought. It has been shown that developing children respond to sign language before verbal language and early age developing children use their own form of babbling and language before adopting formal languages of their parents. Language clearly plays an important part in consciousness and thought, no matter how it is expressed: in verbal or non-verbal form, language is consistent.

The autonomous machine should be able to instruct itself using a Context-Given-Grammar of its own. This is similar to how people think on a conscious level in order to solve problems, often talking themselves through their problems verbally or non-verbally. The autonomous machine need not have input received to product output. An example implementation of this is the autonomous machine writing a script in the Context-Given-Grammar (human-like language) when asked how to solve a puzzle and then executing that grammar script when asked to solve it. From this we can glean the concept that the machines’ language should be what the machine runs in, what it writes scripts in, and how it executes script. The machine’s language should be its core method of thinking. All “thought” should be done in a Context-Given-Grammar in the form of spoken or unspoken language.

The autonomous machine should operate using natural language in the form of self-talk similar to humans.

Programming languages have been shown to be equivalent to finite state machines. A Context-Given Grammar by definition of this paper is a grammar unlike a programming language in that context is preserved during the parsing of the language and the parsing of the language is dependent upon that context. As prior mentioned, markers have been used in chatbot implementations of Neural Networks to provide limits to context for the matter of parsing. It is of the opinion of this researcher that a Context-Given Grammar for an Autonomous Machine must have a similar relationship. The language the Autonomous Machine uses should be congruent to a sort of state machine but of course with a marker for given context. Just as the Context-Free grammar correlates with an instructed machine, a Context-Given Grammar must be paired with a fully autonomous machine. The necessity of memory in maintenance of context will be addressed when we reach the requirements associated with it.

This sounds like a big order, and it cannot be tackled all at once. Requirements are outlined in piece-wise sections and then addressed, integrated with one another, to create propositions for a more complete algorithm and associated data-structure.

**TODO:** Move this to the section on Semantices, Abstraction, and Set Theory

*Conjecture 1.0 Semantics*

In the following usage, *atoms*, refer to the singular smallest units of knowledge. An atom is also a singular *abstraction*, (TODO: INSERT DICTIONARY CITATION OF WHAT AN ABSTRACTION IS). In this paragraph, a *structure* is also an *abstraction* but one that is non-singular when decomposed. It is *structure* not *atoms* that give meaning to *atoms* and produce semantics (TODO: CITE DICTIONARY DEFINITION OF SEMANTICS). As in functional programming and the lambda calculus, it is the structure of the variable, its usage, and the operations performed upon it, that determines the *semantics* of that cluster of *atoms*. An example from arithmetic follows. “5 + 5” is the operation of addition on the atom 5. Whether the atomic operation is “5 + 5”, ”10+ 2”, or “11 + 12” the semantics still lies not in the individual atoms or operations but in the structure.

*Corollary 1.0 Conjecture Semantics*

A *semantic* is analogous to a function. Because an Artificial Neural Network (ANN) can be defined by functions of functions, via recursion (TODO: ADD CITATIONS) semantics can be given to an ANN. To clarify: An Artificial Neural Network, though consisting of functions, can possess meaning.

# System Goals and Structural Requirements

These are the most general and basic requirements for the system, how we hope the system will produce final results based on the entire research conducted, requirements established, data structure(s) and algorithm(s) indicated and the test model that is presented. This is what the researcher hopes to achieve.

**Test Cases**

Test Case A: Have the application show it is able to read a question and search through data (such as search engine results) to find the appropriate and correct answer (for example, a relevant link)

Test Case B: Another test of intelligence is the following: if given a series of instructions, can the system once instructed close to accurately predict the results of the instructions before executing them. This is essentially an advanced pattern matching operation.

Test Case C: The system shall be able to be instructed to solve a task and by nature of its data structure(s) and algorithm(s) be able to output a step-by-step process in a Context-Free Grammar

Test Case D: The system shall be able to be instructed to solve a task and by nature of its data structure(s) and algorithm(s) be able to output a step-by-step process in a Context-Given Grammar (Natural Language that is native to the system)

Requirements:

* All intelligent functionalities must be inherent in and as a result of the structure of the architecture of the system and its respective data structure(s) and algorithm(s).
* The system will be educated by allowing it to data mine articles off the internet for a period of time to populate its respective data structures.
* The intelligent system must generate and demonstrate internal intelligent (human-readable) **self-talk.** This will be revealed through private output of internal processes.

Potential Problems:

* A problem encountered in developing NLP is that computers work off of rigid, restricted sets of vocabulary. For a successful natural computing language, a single object must be able to be referenced by more than symbol set (word or set of words). We hope to cope with this through the integration function that involves the meta-object system and the language of set theory for abstraction.
* Another restriction is for a natural computing language to be successful; it must be read in a flowing manner. The **language must flow like natural language**, unlike current programming languages whose keywords are context-free. This coordination of language output to fit a human language will involve data abstraction and its logical ties to set-theory operations. Through the language of set-theory propositions can be formed and mathematical construction can be reduced to lambda expressions. This requirement is the deformalization of mathematical expressions via the reversal of set-theory constructs and lambda expressions into intelligible speech patterns through a planning mechanism similar to the human brain’s Executive Functioning. This may come in the form of a specific portion of the Meta-Graph data structure.
* A **sorting algorithm** will have to be developed in which a large amount of data in the form of words is input and sorted while corresponding with an **integration algorithm** to retain structured, semantically significant information, without unnecessary repetition (similar to the purpose of the reduce function of MapReduce), retaining not only the meaning of the sentence but its meaning related to other, incongruous sentences. Incongruous sentences meaning sentences not directly spatially located near that sentence. In the case of the Meta-Graph system we recall that sentences are actually the resolved form of local network connections within the Meta-Graph.
* A **disintegration algorithm** that will punish (by definition the antonym of reinforcement) over a measured period of operations or time the significance of portions of the paths in the Meta-Object Graph so that new information can be more easily sorted, integrated, and searched upon. In large data centers this also plays an importance in performance of the network.
* The problem that computers work off a rigid, fixed vocabulary in contrast to humans who work off a flexible, ever-changing vocabulary.
* The amount of processing time and “handling” of objects in order to sort, search, integrate, and disintegrate a complex data structure such as a Meta-Graph in contrast to a simple Map. This researcher hopes the meta-data within the network and the use of a simple logical inference and abstraction language such as Set-Theory will improve these performance hurdles≥

# Algorithmic Requirements for the SELENA Project

The autonomous machine should have three primary modes that are expressed as master algorithm(s) potentially containing subsets of algorithms:

* **Integration** This is what might be called the learning process. In this process the automaton absorbs information from its sensors as well as reflects upon internal states and uses their data to make associations, abstractions, and store them. The Meta-Graph will create new nodes, find similar nodes through search, and integrate those nodes through the language of set-theory and by populating meta-data to link new nodes to old nodes. Learning takes place when the Meta-Graph discovers through **Search** nodes that have corresponding meta-data and is able to create logical inferences to those nodes.
* **Search** This is used for decision making and is similar to what occurs when neurons are fired within the brain. This consists of exploration of paths, which are overlaid over the network’s nodes and arcs. Search algorithms in AI are typically generalized (not including optimizations) as either Depth-First or Breadth-First. Where Depth-First explores down to leaf nodes and recurses backwards and Breadth-First is a broad spectrum search that explores each level of the graph (or in the instance of a network) by node level, iterating more deeply across. Heuristics, markers, and other such optimizations may be used to limit search depth on both algorithms.
* **Sort** In this system this is a cleanup period where data is reorganized and made more efficient for access. Also, processes put on the backburner can be completed during this phase. In the human brain the researcher feels **Sort** most commonly occurs during the sleep cycle, when the brain “resets” itself. This also corresponds to the reduce function in MapReduce that joins key-value pairs of similarity. The optimization of the system falls under the **Sort** algorithm. Re-abstraction also occurs during **Sort**. Large concepts are encapsulated through the language of set-theory into abstract concepts and then associated with other abstract concepts via similar inferences.
* **Disintegration** In this system the query-able meta-data that lives on a node describes its lifetime, the number of interactions that it has had, and its general training over a period. As node reinforcement counters decrease with each overall graph integration, old pathways are handled less by **Sort** and less by **Search**. Paths in significant deficit are removed by **Disintegration** to maintain network size and stability. **Disintegration** is a relatively new addition o SELENA 1 and was added after referencing an article in the Publication of the ACM on Neural Networks and the need to forget.

# Notes on the Correspondences Necessary for SELENA to Implement from the Human Brain for Consideration in Moving Onward

* The brain performs the actions of a pattern matching machine.
* The brain is not only able to remember outside stimuli (such as pictures, sound, touch) but it is also able to remember and generate complex algorithms.
* Impressions are made by the human brain (from the senses) which the brain forms into ideas (an abstraction) according to Hume.
* Recall is some form of unknown search (path finding or tree traversing) algorithm in which associations along the path become recalled as associated ideas within the mind of the individual.
* These unknown search mechanisms would be able to generate and run external scripts. Making some algorithms not necessarily
* Some concepts become heavily weighted and permanent; when used frequently become heavily "weighted" and are remembered easily, appearing permanent.
* **The brain is also able to generate new "ideas" or thoughts.** These are new associations generated by the integration (specifically the sorting portion of the) process.
* We can generalize these mechanisms using terms of language computer science.
* **Our present computer programs, Context-Free Grammars, and Impressions of the Human Brain consist of *Symbols* and *Rules*.** However, a computer program, which is essentially symbols and rules, is written by a human and input into the computer system. In order for us to develop a computer with autonomous intelligence similar to our own, it is necessary we develop a computer system which can **derive its own symbols and rules from its own impressions**. This is re-emphasis on the introduction and what was talked about with regards to the system generating self-talk in the form of a Context-Given Grammar. It will need to derive these from its own impressions. If we can develop a system as aforementioned, it is of the belief of this researcher that we can develop a computer that has autonomic intelligence on par with that of a human. Impressions **must** be part of **Integration**. The system cannot learn properly without self-referencing its sensor input with the current graph to build its new linkages, **even** for totally new information.
* During waking hours, the short-term memory is more active in the human brain, unlike sleeping hours, the brain is receiving minimal outside stimuli. The use of **Sort** during idle time will allow the network to function much like the human brain during sleep, with little stimuli coming in it can perform its reduction and other necessary **Sort** processes. Studies show that the human brain performs similar actions to what is outlined in **Sort** during its sleeping hours.
* Not only must the **Sort** process occur during idle hours like the human brain performs during sleep, but **Integration** process must also go on during idle time as well as active sensor time, just as the human body is resting at night, the human integration process is much more intense, taking the form of dreams. Abstraction of the subnconscious and its respective integration routines appear to be executed during sleep as well when referring to the human brain. A respective clock mechanism similar to the human body clock may be necessary for the network to manage idle and busy times and when data must be cached, integrated, sorted, etc.
* The human sub-conscious from observation and psychological research that has been done over centuries appears to be constructed of abstractions consisting of their long-term memories and their associations that perform an influence on impressions to short-term memory and wake-time integration. The difference between subconscious and conscious may be the reach of the **search** mechanism, with its ability to extend further into the network during idle hours and the amount of abstraction of multiple nodes into single-concept nodes. I.e., the extrapolation of “pain is bad” from events like “I broke my arm and it hurt”, “I hit my head and it hurt”, and so on. These long-term memories under the proposed model would be abstracted during idle hours and their local abstractions would influence short-term memory non-idle time integration.

# TODO: REVISE AND EDIT Sole Requirements for a Purely Autonomous Machine

Requirements for a machine of human intelligence:

1. Must be able to rewrite its own procedures or instruct itself in some other manner (create new ways of solving problems)

2. Must be able to match patterns (recall and recognize information)

3. Must be able to make valid associations (remember information)

4. Must be able to abstract (create more generalized patterns from learned patterns)

5. Must be able to process sensory input and generate sensory output

6. Must be able to learn through rewards and punishment (rewards and punishments are sent via neurotransmitter attachment)

7. Must be able to learn, output, and think in natural language

# Additional Requirements for Selena specifically, as an autonomous machine (TODO: Make sure to revise name at least)

1. The application shall be able to read a question in natural language and

search through data mined (such as a search engine result) to find the appropriate and correct link.

2. Should have all intelligent behavior the direct or indirect result of properties inherent in its architecture and respective data structures and their interaction.

3. The system shall be able to self educate itself through a process of mining data from a large human data source such as the internet.

4. The system shall be able to discern likely correct information from likely incorrect information through recognition of internal conflicts (confusion) with its current knowledge base, whether that knowledge base is supplied or learned in the same manner as the new information.

5. The system shall have if necessary, accessible, ongoing **internal dialogue** in a known and comprehensive natural language such as English.

6. Selena will have to be comprised of three basic functionalities: Search (decision making a.k.a. thinking), Sort (“sleep” or optimization), and Integrate (take sensory input and integrate it into her brain).

# TODO: REVISE AND EDIT Requirements for a Free-Willed Machine

**Definition**

If enslavement is the state and condition where one is forced to follow literal instruction without choice and the will to choose is barred, be it of a set of instructions or not; Then free will is the state and condition where one may choose their action, be it instruction of one’s own generated instruction or of any instruction otherwise supplied to the decision maker, where often an instruction chosen is likened to a path chosen, without the demands of slavery. **Also in order to be self-aware one must have free will.**

1. At the instruction level the unit has the option of making its own decisions

2. At the instruction level the unit has the ability to choose not to follow instructions supplied.

3. At the instruction level the unit has the ability to choose to follow the instructions supplied.

4. At the instruction level the unit has the option (may) choose what it believes best for itself.

5. At the CPU level, the machine has the choice to ignore, follow, or generate its own instructions.

6. The machine may make its own choices, at the CPU level, based upon extraneous data that sways the decision to ignore, to follow, or which instruction to generate for itself. *The machine makes its own decisions based off of current data*

7. The machine may choose options blindly. *The machine can generate an instruction at complete random, essentially choosing blindly*

8. **An autonomous machine must be able to produce output without input**

9. An autonomous machine must be able to query its own purpose

# Requirements for Self-Instruction

1. An intelligent machine that behaves like a human must be able to write, read, and execute its own generated instructions. This is a form of self-instruction for the machine.

2. If these scripts are written at the CPU level and thus consisting of assembly language level instructions, they are to be executed by the CPU as if they were normal instructions input to the processor.

3. In the case of the Selena Automated System, if decision making and instruction occurs on the per neuron level as is suggested by this paper, then script execution and self-instruction must also occur on the per neuron level.

4. If the decision-making is occurring on the per neuron level as suggested above, then it may be inhibited: this decision-making unit (neuron) is asleep, isn’t being use or excited: this decision making unit (neuron) is awake and can follow and generate instructions.

# TODO: REVISE AND EDIT Data Structure Components for Individual Nodes within a Meta-Graph (Neurons)

**TODO:** Only make remarks to neural network construction, path creation and reinforcement/punishment, and leave neurotransmitters out of this particular document for the time being

A neuron in the network is expected to be able to:

1. Read, writes, modify, and execute scripts

2. Trigger script procedures

3. Trigger other neurons

4. Send signals to other neurons

5. Act as a single decision making unit (CPU)

Each node in the network forms a part of a path that is modulated by its core probability value, that is, the value that is the probability that a signal will pass through it. Each node include meta-data that describes it that is supplied by other nodes, so that it can function as part of a set-theory like language of inference and abstractions and not just a simple network of association. The set-theory like language allows for logical constructs to influence signal transmission and help the language portion of the Meta-Graph to construct inference statements in Context-Given Grammars.

# 

# TODO: REVISE AND EDIT Requirements for a Natural Computing Language and its Implementation using Context-Given Grammar within a Meta-Graph

1. For a successful natural computing language, a single object must be able to be referenced by more than one symbol set. A symbol set being a word or set of words in this case.

2. Also a computer natural language must be read in a “flowing” manner, where meaning (semantics) is developed in a **cumulative** manner.

3. The system shall instruct itself using a context-given-grammar of its own, similar to the way people work through problems in their minds using natural language.

4. The machine’s language should be what the machine runs in, what it writes instructions in and what it runs its instructions in.

5. The system shall conduct all thought using a context-given-grammar that takes advantage of natural language as well as the benefits of context-free grammars that exist in present-day scripting languages.

6. The autonomous machine should operate using natural language in the form of self-talk similar to humans. The autonomous machine should make more complex decisions using natural language.

7. The autonomous machine should be able to output and input decisions that must be made requiring assistance from other autonomous entities.

1. A long handshaking process or “context-building” process. A sequence of insignificant phrases is exchanged to gradually build up a context-buffer to work from for more meaningful conversation.

2. The context-buffer that is a buffer of common knowledge between two hosts that establishes the context for the oncoming communication. It should be tailored towards the expected topics of conversation and should occur during the hand shaking process of the CGG.

3. Once an appropriate context buffer has been established, the communicating devices can begin their duty, all the while correcting and polishing the context using more communication until the knowledge in the buffer has reached its necessary level of accuracy (this may be determined statistically according to measurements of time conversing versus subject matter or some other means).

4. Natural language requires a context to be established by definition (whether reading, speaking, writing, or simply listening). This context is built upon both the communicating and receiving’s “common memory” in this portion of the paper called the “context-buffer”. Without it, natural language communication is both useless and not possible.

# Requirements for Search

1. Search must be rapid
2. Search must depend on a rapid access data structure
3. Search must be dependent on association
4. Search should resemble “spreading activation”
5. Search should begin at a single node
6. Search should propagate through the AI, starting at the intial node
7. Search nodes shoujld activate surrounding nodes
8. Search should be weighted, propagation waning as more nodes are activated
9. Nodes cease firing when reaching a certain threshold
10. The interpreting program shall sort the resultse
11. The results are those nodes fired last at the threshold

# Requirements for Sort

1. The system will be required to sort large amounts of data in the form of natural language, sorting the data in a manner in which its semantics are maintained and optimized by location, whose semantics in relation to the surrounding are maintained despite relocation, and who conflicts and errors causing confusion are rectified.

# Requirements for Integration

# Requirements for Disintegration

# Requirements for Abstraction in a Meta-Graph Network

**Definition: Set Theory** (from Dictionary.com)

The branch of mathematics which deals with the formal properties of sets as units (without regard to the nature of their individual constituents) and the expression of other branches of mathematics in terms of sets. (Dicttionary.com)

**Definition: Meta Object** (from Wikipedia)

In [computer science](https://en.wikipedia.org/wiki/Computer_science), a **metaobject** is an [object](https://en.wikipedia.org/wiki/Object_(computer_science)) that manipulates, creates, describes, or implements objects (including itself). The object that the metaobject pertains to is called the base object. Some information that a metaobject might define includes the base object's [type](https://en.wikipedia.org/wiki/Type_system), [interface](https://en.wikipedia.org/wiki/Interface_(computer_science)), [class](https://en.wikipedia.org/wiki/Class_(computer_science)), [methods](https://en.wikipedia.org/wiki/Method_(computer_science)), [attributes](https://en.wikipedia.org/wiki/Attribute_(computing)), [parse tree](https://en.wikipedia.org/wiki/Parse_tree), etc. Metaobjects are examples of the computer science concept of [reflection](https://en.wikipedia.org/wiki/Reflection_(computer_programming)), where a system has access (usually at run time) to its own internal structure. Reflection enables a system to essentially rewrite itself on the fly, to alter its own implementation as it executes.

**Explanation of Requirements**

Set-Theory provides a language to perform abstractions without the need for concern of its individual components, that can be used as a foundation to prove all system of mathematics and logical inference. Integration of this into a neural network seems ideal to apply the operations of mathematics and logical inference to the mechanisms of neural networks as well as a built-in method for abstraction. Inference as of right now is a popular topic of implementation in Neural Networks and in-regards to Deep Learning. By implementing a meta-object system in the context of the language of set-theory we hope to improve our Neural Networks to perform abstraction more like the human brain does.

1. The system shall support the abstraction of concepts
2. The system shall support set theory as the basis of mathematics and logical inference
3. The system shall use set theory constructs and th4e language of set theory to perform abstraction f the network
4. The system shall use the concept of meta-objects
5. The system shall use the meta-objects to self-describe abstractions

# High Level Details of SELENA’s Architecture

Many of the sections listed below can be implemented using a semantic network, neural network, semantic-neural network, or the Meta-Graph data structure described specifically for this paper, with each individual network communicating to the others as listed above (i.e., sound encoder->sound pattern matcher->main pattern-matcher, etc.)

**Sensors (Raw bits)**

The role of the sensors is to provide raw data. The most external portions of the Meta-Graph.

**Encoders (How is it?)**

The encoders pull information from the sensors and encode it into a universal language for processing consistent with the data structure being used (in this instance a Meta-Graph). Note that when data is encoded, it includes the data type (visual, auditory, or tactile) so that it can be categorized for storage and for delivery to the conscious peak. Encoders could be implemented using normal scripts or as an ANN type system, trained to encode natural language into ANN structures. In this specific case, encoders are part of the Meta-Graph and function as populators of meta-data for the graph.

**Sensory Pattern Matchers (What is it?)**

The primary sensory pattern matchers communicate with the encoders and the primary integrated pattern matcher. Communication flows from the instructions of the primary integrated pattern matcher to the primary sensory pattern matchers, down to the encoders and also in the opposite direction. The encoder takes the encoded data in a form where it can be pattern matched (from visual bits to a small ANN, semantic graph, ANN-Semantic Graph, or series of token strings genrally) and pattern matches it with other sensory data of that type that has occurred previously or is engrained “genetically” in the system. For example, visual encoder captures the bits that make up a ball in its vision. There is also a giraffe and a boulder in its vision. This data is passed to the encoder which encodes it into some sort of visual data structure representation, that is passed to here (the sensory pattern matcher) which identifies the objects as a ball, a boulder, and a giraffe. That data is then passed on to the primary integrated pattern matcher. Just as the other portions of the system, the sensory pattern matchers are formed as specifically tasked Meta-Graph components, just as the human nervous system uses neurons and bundles of neurons to accomplish these tasks.

**Primary Integrated Pattern Matcher (What to do with it?)**

The primary integrated pattern matcher communicates with both the primary sensor pattern matchers, the memory unit (the central portion of the Meta-Graph), and the conscious peak which is used for direct input/output operations on the short term memory unit. The primary integrated pattern matcher serves as a sort of traffic cop, directing flows of information from the internal memory and to it. It plays a part in decision making. It records patterns (abstractions) from its impressions (sensory input) and recalls them. Following the example from above, the primary integrated pattern matcher would receive the signals for “Ball”, “Boulder”, and “Giraffe”. Taking those signals, it would match them against signals in the conscious peak (which includes short term memory and context), and long-term memory (which exists in the memory storage unit). The PIPM may contain abstract constructs (such as ball, boulder, giraffe, horse, etc.). Those abstract constructs are built into the network (be it a semantic network or neural network typically, in this instance it is a Meta-Graph). Each abstract construct may be permanently connected to events in the memory storage unit (long term memory). It also may be temporarily connected to subjects in the conscious peak. The PIPM, when pattern matched (by either the pattern matching units below [incoming senses] or the conscious peak above [recall]) will match against abstract concepts it is familiar with, and if there is a match, the process will flow and continue on to pattern match against the long-term memory network, thus recalling those memories.

On some levels, the PIPM also performs decision making, so called “subconscious” decision making.

**Memory Storage Unit**

The memory storage unit is used to store long-term memory type data. It exists as the frontier of the Meta-Graph, nodes that are typically unnecessary to reference during search or nodes that have been abstracted by sort.

**Conscious Peak**

The conscious peak is the portion of the pattern matcher that is “aware” of its own processing. This includes functional concepts such as “self-talk” and “conscious decision making. This consists of the most active portion of the Meta-Graph. It is the nodes triggered by impressions that have the most immediate search “radius” according to the search markers. Search markers are used to limit search as a heuristic for Context-Given Grammars and Natural Language Processing.

# Neuron Clusters : Cores

This is another way of organizing or “looking at” Selena’s brain which does not invalidate previous descriptions and definitions but supplements it. It also includes other information not included in other details of the model of the system. The *Action Core* would contain sensors, encoders, and pattern matchers for physical processes. The *Sensory Core* would contain sensors, encoders, and pattern matchers for Selena’s senses, and the *Cognitive Core* would contain all pattern matchers for cognition, the handling of the other two cores interoperability, and essentially the main processing portion of the brain. The *Cognitive Core* is responsible for the central mapping of all other cores, sensors, and effectors. It is the portion that is the primary abstractions, it handles most dense interoperability.

This core organization must also be considered when architecting the layout of the Meta-Graph data structure(s).

*Action Core*

Contains neurons associated with actions. Physical actions in human beings. Gross and fine motor skills, etc. Fringe nodes.

*Cognitive Core*

Contains neurons associated with “thought”. Language generation, planning, organization, dense mapping and interoperability of other cores, self-instruction, prediction. This core of the synthetic neural “tissue” or network is the center of the network, hence the density. It is responsible for mapping signals from all other cores to each other. By mapping one form of abstraction to another the Cognitive Core can associate things like the visible word Giraffe, with the picture of a Giraffe, the sounds of the word Giraffe, and for example, the feeling of a Giraffe’s skin, the smell of a Giraffe, or if we ate Giraffe meat the taste of a Giraffe.

*Sensory Core*

Contains neurons associated with the sensors. Physical senses in human beings. Fringe nodes.

Information is automatically “sorted” into these three cores and their sub-cores. In humans, genetics determines the sorting algorithm. Information is automatically “sorted” into these three cores and their sub-cores. In humans, genetics determines the sorting algorithm used to spatially locate these cores in the brain for chemical and signal optimization purposes.

# Meta-Graph Data Structure in Entirety

**TODO: Flush out fully using set theory (also reference LISP and Lambda Calculus)**

The abstraction logic language used to construct the Meta-Graph (detailed in another paper) should have its operations non-binary, their operators should be the thing tuned by the system, the operators on the abstractions (sets) are what are weighted to influence the signaling of the network. Therefore, the operator weights control things like “inference”, “unions”, or “intersections” of abstractions.

TODO: All classes should inherit from MetaObject, which contains a member Meta-Data (so I don’t have to list that instance in every class)

TODO: All data members are forms of sets (or atoms). All classes also have a lambda element that can be used as a function according to lambda calculus?

TODO: Somehow show how probability reinforcement is counted for algorithm purposes. (The markers that need to be used for search, what resides in long term memory and short term memory, what is to be accessed by natural language, etc.)

TODO: Operations such as lambda calculus functioions, first order ogic operations, and possibly second order logic need to be somehow embedded in an asbtract and fluent way. Research first-order logic, second-order logic, and lambda calculus. There must be way to include operation as well as data in sets that makes the most sense for the meta system.

TODO: Figure out what algorithm should be used for the hint system. This is the biggest hurdle right now. If we look at the forwarding action of logic operations we might be able to find a way to provide a look-ahead system to provide meta-hints. The look ahead system could operate over abstractions instead of individual components, increasing efficiency.

**Class GraphList** //an unknown data structure with an unlimited list that is efficient

Instances of Atom

**Class Meta-Data**

Instance Meta-Data

Instance Set

GraphList (contains an unlimited amount of data in any type)

**Class Set**

***Data Members***

Instance Meta-Data

Instance Atom

**Class Atom**

Instance Meta-Data

Instance Set OR Instance Bytevalue (Primitive – could be an int, a string, virtually anything) OR Instance Operator

**Class Operator**

TODO: Lookup Set Theory Operators

Instance Union OR Instance Intersection OR Instance Difference.

**Class Node**

Instance Soma: Class Semantic-Arc //output

Instance Dendrite: Class Semantic-Arc //input

Instance Axon

Instance Meta-Data

**Class Soma**

Instance Weight (Probability-Score)

Instance Meta-Data

**Class Dendrite**

Instance Meta-Data

GraphList<Node>

**Class Axon**

Instance Meta-Data

GraphList<Node>

**Class Semantic-Arc**

Instance Meta-Data

Instance Set

**Class Weight**

Instance Meta-Data

Instance Set

(STOPPED HERE FOR THE NIGHT)

TODO: Edit Text Below

Above is a generic example of nerve cell. The tree like dendrites at the top receive electrical signals, that pass through the soma that is the heart of the nerve cell, and are output along the single axon into another dendrite through the synaptic gap, onto another neuron. Neurotransmitters exist in the synaptic gap and carry the message from neuron to neuron.

In the case of this data structure, each, single, node contains a binary state that represents the node among other nodes through its weights. There are probabilistic weights on each axon and on the many branches of the dendrites. Both the dendrites and the axons are represented by ‘edges’ in a graph, with the soma’s being the ‘nodes’ or ‘vertices’. Each soma+axon+dendrite triple along with other triples, form a path. Each path has state and can represent various things, from the color blue to the Disney film Snow White, depending on how complex the path is.

**Weights**

When the potential coming through is less than the probabilistic weight of an edge, that edge will not be passed. However, if all the weights on a neuron are high enough that no edges are passed, the potential (“voltage”) in the soma will accumulate until it finally can be let out over one of the weighted edge, the one with the most certainty will be crossed first.

# Sort (Reduce) Algorithm

# Search Algorithm (WIP)

General Algorithm Purpose:  
Take an input signal

Generate an impression – an impression is a mapping of the input signal to present network nodes and pathways a subset of the network

Perform this search using some sort of Depth First Search with a global marker to reduce processing-time and over-signaling. Marker is like a dynamic heuristic.

Perform depth-first search where necessary to explore deeper parts of the network where necessary

Algorithm must involve both breadth-first search and depth-first search ijn order to properly explore the network

The algorithm might want to use meta information on each node to perform search in some way

The algorithm may want to use set theory operations (abstraction and higher-order logical operators) when performing search

Abstractions most likely should be used to represent general search results, with more defined searches of the sub-elements of these abstractions (represented through sets) necessary for further matching

The main problem is, how do we pick the start node to begin search in the network?

Search should return a portion of the network data in some format, either abstraction or some other data structure that is a subset of the network.

Search should probably return a result that can be converted to natural language

Search should return a result that is logically connected

Given input signal X

Update global static timestamp variable

**General Algorithm**

FUNCTION SEARCH( NODE X, SIGNAL Y )

CACHE (MASTER TIME LIMIT MARKERS)

WHILE ( CACHE ISNOT EXHAUSTED )

IMPRESSION = MASTERSEARCH ( NODE X, SIGNAL Y )

END WHILE

RETURN IMPRESSION

END FUNCTION

FUNCTION MASTERSEARCH( NODE X, SIGNAL Y )

WHILE (CACHE IS NOT EXHAUASTED)

INITIALIZE BFS\_LEVEL  
 PERFORM BFS ( NODE , BFS\_LEVEL )

END WHILE

END FUNCTION

FUNCTION PERFORM BFS( NODE, META HINTS, IMPRESSIONS )

QUERY BFS NODENEIGHBORS FROM NODE

LOOP THROUGH NEIGHREST NEIGHBORS

GET META HINTS FROM NEIGHBORS NODES

IF NODE META HINTS MATCH NODE

IMPRESIONS->ADDTOWITHMOREWEIGHT( NODE )

ELSE

IMRESSIONS->ADDTOWITHLESSWEIGHT( NODE )

IMPRESSIONS = PERFORM BSF( CURRENT NODE, NODE->META HINTS, IMPRESSIONS)

END LOOP

RETURN IMPRESSIONS

END FUNCTION

FUNCTION MATCH HINTS

//will need to perform an assessment of incoming metadata on a node

///an examined node to see if there is a correlation

//this should be done by seeing if logical inference occurs to indicate a

//similar train of thought or if abstraction of similar concepts occurs

# Integration Algorithm

The general goal of the Integration algorithm is to, given a set of inputs, integrate those inputs into the rest of the system. Easier said than done. The Integration algorithm must be able to take input (source) impression data, find similar mappings to it in the abstraction portion of the Meta-Graph, the destination impression data, and then tie and create nodes in the Meta-Graph that are linked to those related nodes and associations. It does this by re-abstracting already present data so that it links to known abstractions. This is the idea of “I can relate to what you are saying.” In this process of developing a relationship between source and target impressions, it has to adjust the weighting of the somas of the neurons in the Meta-Graph to reinforce, or punish the already present impression data by the source data.

In technology, computers, communication, psychology, and many other fields there is the idea of protocol and handshaking. Take two people from different cultures speaking different language stalking about the same thing but unable to communicate. They perform a handshaking operation and use protocol, that is, there are common procedures to go about taking source impression data and receiving it as target impression data. This common procedure sometimes is pointing, sometimes it is other common noverbal communication. Eventually a common communication protocol is established before more detailed impressions can be exchanged. This handshaking process can also be accomplished when teaching by cross-reference. A small seed is planted in the system, and as new inputs enter in, they cross-referencable symlinks form. As the system builds upon itself, it “learns”, it is able to cross-reference more data.

This is important in all communication. Even among family members who speak the same language and know each other deeply. The exchange of impression data via cross-referencing is what allows for common topics of interest to negotiate relationships and build common language to exchange ideas.

# Disintegration Algorithm

# Experiments and Results

# Sources (Citations/References)

# TODO: REVIES AND EDIT Notes on Machine’s Context Given Grammar and Context

**More notes on a Context-Given-Grammar**

In order to make a context-given-grammar effective, a long handshaking process must be involved. The common language being used has as a main portion to it a large “context buffer” consisting of common knowledge between the two that establishes the context for the oncoming communication. Once an appropriate context has been established in the buffer during the boostrapping process, the communicating devices can begin communicating naturally, all the while correcting and polishing the context using more communication. The longer the two devices are separated, the more time is necessary to re-establish a communication context and the larger the buffer to be filled.

Natural language is an exchange of information by definition and by definition requires a context be established. This context is built on the communicating devices own memory and also on memory of the device it is communicating with, known as “common memory” or “common ground” as we would call it. Without common memory natural language is both useless and impossible.

# TODO: REVISE AND EDIT Behavior and State vs Structure

An M-Network like what is shown above has two different sets of information integral to it. First is Memory. By storing a soma named “Alice” and a transition function “went” along its output edge (consisting of an ouput Axon and an input Dendrite) to another soma “Store”, we can record very detailed information about an event in memory. We can continually build on the events of Alice and the Store. But, since there is no state in this data structure we have built that is essentially just a graph, we need to keep another set of data. This other set of data is the “path” data. With each path actually corresponding to a memory, that memory could be data (pure state) or it could be a program (pure behavior). Thus, in the case of a program, the path is actually a set of encoded instructions.

Therefore, in an M-Network, which is the name I have given to the above data structure, you first must lay down the grid of the nouns and verbs, (the what’s and how’s) in the form of vertices and edges, and meanwhile you must draw along those graph edges when the sentences have been used.

# TODO: REVISE AND EDIT Thought versus Memory

Every time a pattern is put to match against other pre-existing patterns, new nodes are created to represent and store the new pattern should it come up again. These nodes are connected to the pre-existing nodes. For example:

I) Match the phrase “A tree fell in the forest” to “A tree fell over the freeway”

Incongruous nodes: over the freeway : in the forest

II) Search for nearby nodes for the word “forest”. Treat “fell”, in”, and “the” as part of an arc (or edge or whatever you want to call it).

III) Found no match locally (distance parameter set by the programmer of the network) for forest. So, creates a new node called forest and links it to “A tree” via another arc, “fell in the”. A tree has directed edge called “fell in the” that now points to forest.

IV) Create a temporal (instance) path that traces itself through “A tree fell in the forest”.

# TODO: REVISE AND EDIT Mathematical Definition of an M-Network

*An M-Network is short for Meta-Network, it is a network constructed of Meta-Datathat allows for to be easily searched, restructured, and recursed. It evaluates to an algorithm, and when combined with Arkhum’s Conjecture, evaluates to a series of self-describing differential equations.*

Let there be a set of paths, P = { p0, p1, p2, …, pn }

Where each path pi corresponds to a concept.

Let p be a set of vertices and a set of edges, p = { d, V, E }, V = { v0, v1, v2, …, vn }

E = { e0, e1, e2, …., en }

Where d is the degree of the path. Zero by default. Increases by one

for each level of hierarchy. Paths may only consist of vertices and edges with the same d degree of hierarchy. Paths are part of the static structure of the network data structure.

Hiearchy example:

I got a PC I, got, a, PC are nodes

/|\

g o t <- g, o, and t are nodes

Let each vertex v be a set, v = { S, [p] }

Where S is a string in the form of a noun representing some form of state

And [p] is an optional path, allowing paths to be treated as individual nodes to be connected together on a hierchical graph.

Let each edge e be a set, v = { R, in\_r, out\_r, w, τ }

Where R is a string in the form of a verb representing some form of state

Where in\_r, and out\_r are references to the in-vertex and out-vertex of the edge

Where w is a weight no less than 0 and no greater than 1 that represents the

probability of the state being true, and inversely, the resistance to transition for the signal over the edge. (The higher the probability the lower the resistance).

For every graph there is a τ the transition tuple. τ = ( ρ0, ρ1, ρ2, …, ρn, ε0, ε1, ε2, …, εn ) representing a dynamic path through the graph, where ρi is an element of the set V from above and ε is an element of the Set E from above. Tau is allowed to be directed and cyclic. In this case, the function of each neuron ρand its edges ε function similarly to nodes and transition functions (arcs) in a finite state machine. However, state information is still stored in the semantic-network style ontology.

# Artificial neurons as processors, instructions, and node level decision making processes.

If we are to reduce the cognitive processes of a computer in comparison to a human being, reducing the decision making processes of a machine results in assigning every singular action to the cpu of the machine following a simple instruction that is a part of the machine’s cpu instruction set, In the case of the human brain, the mechanism for “free will” or decision making is unknown, if it even exists. Either way the mechanism for configuring a machine to form its own decisions would have to be embedded within the machine’s instruction set and CPU (which executes those instrutions).

The CPU in the decision making system would have to base its decisions off of known data, and possibly in a binary fashion (either good for the organism or bad for the organism). This decision making process may occur on the neuron level (or pseudo-neuron level as in our machine).

Given an input of predetermined instructions to a processor, unique decisions may not manifest with high probability, that is, predetermined outcomes will be computed. However with every action in a human being, a unique decision is the result of “CPU” outcomes. This would not be possible in the sheer quantitiy in a pure stochastic-emergent behavior system. The probability as mentioned earlier is not cohesive with this result.

*Conjecture*

*Different Neurons firing results in different behavior. For example, the neuron with state “Hello” as a part of “Hello Sam”, when fired, will cause the output “Hello”. Groups of neurons or paths of neurons can fire and trigger other paths or groups of neurons as well.*

*Planning ahead seems to be an important part of the decision making.*

*The neuron on the far left fires, triggering the next couple, which trigger the next few, which trigger the next many, etc.. until almost all have been fired (except for those who are inhibited as the one above marked by [] is). The one neuron decision maker fires to other decision makers, who need other neurons ‘opinion’ on either the good or bad of the decision, until finally the entire brain has been used to decide whether the decision would be good or bad, each neuron making a decision and giving its input.*

An example of this is the following:

Suppose my mind receives the signal that I am hungy and to eat an apple. I then make the decision to eat the apple or  not to. In a traditional machine the instruction to eat an apple would simply be received and then executed. What part of us makes that decision and how does it work in contrast to a simple CPU??

Selena's architecture is to be designed in a hybrid intelligent system, where the upper layers are used for logical and abstract reasoning (mainly production rules), and lower layers are sub-symbolic

Standard PC CPU Instruction Set:

ADD

SUB

MOV

DIV

MUL

AND  
OR

NOT

Possible Data Flow for Making a Decision:

Sensor to Context Buffer to Pattern Recognizer to Recognized Node or Node Paths to Conscious Mind as Brain “Lights” up from adjacent noninhibited neurons firing, decision is made.. As Brain “Lights up” thoughts take form from abstract association to subconscious symbol manipulation to unconscious self talk to conscious (controlled) self talk to behavior (such as speaking or moving).

***Missing:***

*Instruction Set for a Decision Making Processor*

*Internal Logic Gate Structure for a Decision Making Processor*

# TODO: REVISE AND EDIT Actualization

The machine on the high level must, like humans, develop goals and fulfill them. By developing goals the the machine learns how to teach itself. This is one of the most important aspects of a developing AI child.

# TODO: REVISE AND EDIT Testing and Experiments

**Experiment 1**

For the purpose of this initial experiment on the Neural State Machine, we will not be using the recursive hierarchical tree structure and instead use a flat series of node. The goal of the first experiment is to see if we can consistently generate “self-talk” within the “brain” made up of a Neural State Machine by continually stimulating individual, different neurons. However to do this, we must first determine the way in which the brain is rewarded, thus solidifying certain weighted edges over other weighted edges, how the brain receives the stimulation (does not have to be complex at this point) and how and what initial data to use.

I’m hoping in this first experiment, the Neural State Machine I am calling Natalie will be able produce some sort of emergent “self-talk:”. Some

for the future would include developnig a sort of BIOS system for Natalie so one would not have to take her down or go behind the ‘façade’ to directly command her if necessary (for things such as shut off, or diagnostics, or fast training etc.). It would be nice to have Natalie run as service eventually, so that she need not be turned off or explicitly launched. Also, finally in her code I would like to implement the ability for her to use a scripting language, in this way she would be able to write her won scripts and run them, forming a sort of circle (Natalie has exposed methods for doing things, Natalie can write scripts to combine these exposed methods, Natalie can run those scripts, all the while the core code stays intact.) With the ability to run scripts she may even be able to manage certain available tasks such as email management, etc… assuming methods for interacting with the he base os are exposed. Lastly, a working cache memory for keeping context may allow Natalie to speak fluently and speak gracefully, instead of simply outputting words or sentences with little meaning to past output.

Note, my login phrase for Natalie is “Good morning beautiful”

--

The brain has to start off with something. After launching Natalie, it will idle until you give it the command AGE BIRTH to create a fresh brain or BRAIN LOAD to load a stored brain. In initial mode the brain will accept everything coming in as a new node, and will not pattern match until the BIOS command AGE TODDLER is given.

For the purpose of the first experiment and looking for self-talk, we will input several sentences in AGE BIRTH and then set UTILITY TELEPATHY ON to see what is going on inside Natalie’s head. (self talk, emergent behavior, or just gibberish). Also, UTILITY BED TIME STORY can be used to read a large amount of text, like a bedtime story without cutting and pasting from the file. Note, all communication, whether it be inward self talk or outward speech, is regulated by a timer so it does not move excessively fast and overcome either the user or the computer.

--

With AGE TODDLER activated, the brain will begin pattern matching and adjust using rewards/punishments. If an item input cannot be pattern matched, it is stored as a new path. Our goal at this stage is to have Natalie act similar to human toddler or older.

The brain is essentially a pattern matching machine.

Impressions are made by the human brain (from the senses) which the brain forms into ideas (an abstraction) according to Hume.

We can generalize this using terms of language and computer science.

Essentially the brain consists of two things, symbols and rules. This is very similar to what computers use as 'programs'. However, a computer program, which is essentially symbols and rules, is written by a human and inputed into the computer system. In order for us to develop a computer with intelligence similar to our own, it is necessary we develop a computer system which can derive its own symbols and rules from its own impressions. It will need to derive these from its own impressions. If we can develop a system as aforementioned, we will be able to develop a computer that as intelligence on par with that of a human.

Sleep consists of an intensive integration process of data internally stored in the brain and

in short term memory, but unlike waking hours, the brain is receiving minimal outside stimuli

The integration process of data is a biologically based SORTING algorithm the brain

uses to evaluate assocations, determining their validity or invalidity, and restructure

assocations. The integration process goes on during the day as well as night, but as the

body is resting at night, the integration process is much more intense, taking the form

of dreams.

Recall is some form of unknown search (path finding or tree traversing) algorithm in which assocations along the path

become recalled as associated ideas within the mind of the individual.

The brain is not only able to remember outside stimuli (such as pictures, sound, touc) it is

also able to remember and generate complex algorthms. This is similar to a computer program

that would be able to generate and run external scripts. These algorithms are not necessarily

permanent, but when used frequently become heavily "weighted" and are remembered easily, appearing permanent.

The brain is also able to generate new "ideas" or thoughts. These are new associations generated by the integration (specifically the sorting portion of the) proc

# TODO: REVISE AND EDIT Input and Output on semantic and syntactical levels

The primary problem related to both semantic networks and the semantic web in general is NOT entirely an issue of NLP (processing and interpretation), but involves representation of the language semantic data in the form of storage and data.

With further study of both ANN, the way the brain stores information, language formation and usage,etc.. the issue of representation may be more easily solved. It may require some sort of probability incorporation or fuzzy logic to work though.

This is a description of primary problems in no particular order of relevance:

**1. Problem 1: Synonyms**

There are many words that humans use which are are synonymous. One potential problem related to this is the network overload that would occur due to potential loops caused by synonyms which is also loosely related to size and speed problems on the network, another major issue.

**L**

**2. Problem 2: Network Overload in general**

With the amount of data we are hoping to represent (for instance even the amount of data in a human brain) we would have to develop an excellent storage system beyond what we have present day.

**3. Problem 3: Relationship Phrasing**

This I believe is the most significant problem of all and is the problem I have been trying to tackle for years.

The essence of the relationship phrasing problem is that computers use concrete data and logic when performing behaviors and representing logic (and thus representing relationships between data). The OO concept of data abstraction has pulled us away from this slightly, but not nearly enough for an effective semantic network data structure and data algorithms to work in a fashion similar to our brains.

Because of the neuroplasticity of the brain, we know that many parts of the brain can take over for other parts. An individual can have an entire hemisphere of their brain virtually unusbale and still function at a somewhat acceptable level with therapy and medication. There have been cases of this occurring. The brain has a strong ability to adapt, something unfortunately computer programs are **strongly** lacking.

For example, let us try and simulate, using propositions, how we would store some data that a human might have represented in his brain:

*Phrase: My professor is named Dr. Kaplan*

I am using | to separate phrase parts of speech (subjects, verbs, and direct objects), and {} to denote quantifiers

I | have a | professor

Dr. Kaplan | is a | professor

{Some} professors | have a | Ph.D.

{Some} professors | have a | M.S.

{Some} Dr.'s | have a | Ph.D.

{Some} Dr.'s | have a | M.D.

Professors | when | Drs. | have a | ph.D. (compound proposition, more than one implication)

I | have | an arm

I | have | a degree

I | have | an M.S.

Note the fact that I have an M.S. does not make me a professor making this a single (one way) implication. Thus it is evident that we must distinguish when processing NLP between whether a statement is a simple implication or double implication (bi).

Also note above, that the usage of "have" for having an arm is much different from "have" for having a degree which is different for "have" for having a professor. There is a context dependence, which the brain distinguishes based off of prior experience, or prior contextual usages, illustrating that the brain most likely uses a weighted system of connections (allowing for altering of the weights according to prior experience) to determine the meaning of the word "have" in that sentence.

Because of the imprecise nature of human language, it would typically not be deemed practical to store this information isolated in a computer system, and would make more sense to store it in the form of propositions...right? But if we do this we run into that major problem, attempting to translate Human Language, which is heavily context dependent and dependent upon background environmental (including cultural) knowledge into something which is completely discrete and

concrete. This is, basically, impossible and would most likely be easily to prove to be impossible. (Go more into detail about this) (The proof would involve the idea of containment, and in that as the amount of information being represented increases, the amount of contextual information needed to represent it increases at a higher rate. By the time we get to the amount of information stored in a human brain, we would need an incredible amount of context information.? Not sure about this, but my intuition leads me to believe this would be either NP Complete or completely impossible under today's hardware - especially when trying to accomplish this in regards to the semantic web)

So, computer scientists have tried several solutions so far. 1) Use metadata to reduce the context dependence. 2) Translate the statements into discrete propositions ahead of time 3) Restrict the usage of language to be translated

I hate to be critical, but essentially these "solutions" are not solutions, they are similar in mind to the old adage of trying to fit a square peg into a round hole. Sure, you can do what they are doing, and try and carve off the sides of the square so it fits, or carve out the hole so it fits, but wouldn't it be just easier to find a round hole to put it in?

We know from the above analysis and discussion that we need a system developed that can efficiently, accurately, and quickly store and retrieve natural language. To try and do this with a relational database or traditional data structure brings us back to the square peg round hole adage.

In order to store natural language we will need at least four major components:

1. Subjects

2. Verbs

3. Direct Objects

4. Quantifiers

Other proposition components (such as using verbs (connectives) as implication or bi-implication) would have to be completely context driven and based off of weights. Thus, the meaning of "if" as implication and "is" as implication or bi-implication would have to be based off prior experiences or context figured out intelligently. The same with "have" as seen above. The sub data structure representing both subjects and direct objects should be interchangeable and could take the form of a node (like in a tree or graph structure) possibly.

Another requirement of this complex data structure is it must be self-modifying. That is, it must be exhibit similarities to the brains own neuroplasticity.

**Problem of Adjectives and Quantifiers:**

*Need to solve how adjectives and quantifiers are used together and within the data structure*

**Problem of Information Extraction (Inferencing/Drawing Conclusions)**

*In this case both output and inferencing would function identically, unlike in traditional systems where inferencing is only used for proofs.*

*Need to solve*

**Problem of actual architecture of the data structure and modification**

*Need to solve*

**Final problem (related to the last):**

In order for us to develop a fully contextual based system of storing, understanding, and inferencing using natural language there is one other related problem we must tackle. Unlike in a graph, neural net, semantic net, etc.. the human brain is able to broad spectrum pattern matching. For instance, the brain can decide the usage of the word "have" as seen in the example above, by immediately referencing and weighing every usage of the word "have" used before and thus making a decision on how to store the incoming proposition. No matter where the word, "have”s usage was stored in the network. Thus, the following structure won't suit:

I----have-----a dog --- is a --- canine

|

|

have

|

|

a cat

Essentially, every single connective must be wired in a quick way to every other connective. Thus, although the connectives can be used in more than one proposition, there can only be one instance of them in a feasible data structure. (Having multiple instances of them interconnected would be infeasible for the model being presented...at least that is my thought as of right now. The only way that would work is if all the "have" connective nodes would be connected together and have a weighted connections to their respective subject/direct object nodes. The input processing portion of the algorithm would then have to evaluate all the subject/direct object node weights, decide which the input is relevant to, and then form new connections to all of these old subject/direct object nodes. Not to mention if we use this approach, there is no way to really to use weights in regards to the connective nodes as we have created multiple nodes for every connective, thus every proposition has its own unique connective node instance). Therefore the only conclusion I can come to is that we need have a unique node for every subject/direct object type, and a unique node for every connective type, and weighted connections between. Although this hardly fully expounds upon the problem we are faced with.

**Problem of Unknown Input (Input that cannot be directly associated with any prior information):**

The "Memory" problem, or Network Overload Problem. That is, having so much information, that the weights are not being balanced correctly. This can result from either too much information being output (everything seems equally related because of the weights), or no information being output at all (the weights have all been reduced to 0 or are somehow conflicting) . One solution for this problem (besides finding an optimal weighing algorithm) is to have the data structure have a forgetfullness over time. That is, a separate weight will be given representative to time (which is used int he proportional factor of the relevance of the data). Therefore, unless the data MUST be retrieved for some critical decision factor, the data structure will eventually disregard it.

**Possible Solutions to the problem of synonyms in regards to the data structure**

Obviously, if we require a condition of uniqueness for the nodes, it follows that having multiple nodes of the same meaning could present a problem. (Possible solution, storing synonyms in the same node?? - not sure if this is a good idea)

**Problems with words with more than one meaning and uniqueness:**

# TODO: REVISE AND EDIT Neuron Node Path Data Structure

The above drawing represents a node in the graph, with the top arrow showing the single input to the neuron node and the arrows leading away representing the directed output edges to other neuron nodes. The stars are present to represent the existence of neuro-transmitter global states that activate different groups of neurons dependent upon their presence.

Each node contains a binary state that influences the other nodes through weighted edge connections. The sum of the weighted connections may not exceed one nor be reduced less than zero, as they represent the probability that edge will be traveled when the node has been activated.

Neurotransmitter states exist within the data structure construct. Different neurotransmitter types represent different global states. Any neuron may only connect to other neurons through the usage of a neurotransmitter from the neurotransmitter pool.

There exists a set of paths P, with each path corresponding to a concept, that is, the data stored from the path acts as semantic data. Each node holds a collection of individual states which represents broader semantic data. For example, there may exist three nodes, one that represents the state of g, o, and t. Together there exists a path in the set of paths through the states of g, o, and t (symbols) that represent the word got.

There also may be a set of paths of paths, such that the paths of the words I, got, an, a exists, representing the sentence I got an A. This continues on, growing from a sequence of letters to suffix/prefix/bases to words to sentences to paragraphs, and

so on. The data structure that represents this I call a “recursive path tree”, with the nodes of the tree representing paths, paths of paths, and paths of paths of paths.

Therefore the initial states of the recursive tree of paths is a single path. And a single path is made up of a sequence of references to edges. There exists a nonempty set of vertices (or nodes) V of which every edge E consists.

There exists a set of edges e of E that contains a reference to an in-node a(end ‘state’) and an out-node (initial ‘state’). Each edge has a weight w associated with it, with all of the weights w for the node adding up to no greater than 1.0 and are no less than 0.

For every path there exists tau, where tau is an ordered tuple of functions, for edges, each representing the path through the graph in the form of an initial node, directed edges, and accept node (end node). Tau represents the set of transition functions. P represents the physical path that is overlayed over the static data structure.

The data structure is divided into two components, the static and dynamic. Static components include vertices(nodes) and edges. Dynamic components include weights and paths on the edges.

In this case, the structure of each neuron corresponds somewhat similarly to a node/vertex in a finite state machine and its edges corresponding to the transition functions , however state and operation information is stored similarly to an ontology at the same time.

**Experiment 1**

For the purpose of this initial experiment on the Neural State Machine, we will not be using the recursive hierarchical tree structure and instead use a flat series of node. The goal of the first experiment is to see if we can consistently generate “self-talk” within the “brain” made up of a Neural State Machine by continually stimulating individual, different neurons. However to do this, we must first determine the way in which the brain is rewarded, thus solidifying certain weighted edges over other weighted edges, how the brain receives the stimulation (does not have to be complex at this point) and how and what initial data to use.

I’m hoping in this first experiment, the Neural State Machine I am calling Natalie will be able produce some sort of emergent “self-talk:”. Some goals for the future would include developnig a sort of BIOS system for Natalie so one would not have to take her down or go behind the ‘façade’ to directly command her if necessary (for things such as shut off, or diagnostics, or fast training etc.). It would be nice to have Natalie run as service eventually, so that she need not be turned off or explicitly launched. Also, finally in her code I would like to implement the ability for her to use a scripting language, in this way she would be able to write her won scripts and run them, forming a sort of circle (Natalie has exposed methods for doing things, Natalie can write scripts to combine these exposed methods, Natalie can run those scripts, all the while the core code stays intact.) With the ability to run scripts she may even be able to manage certain available tasks such as email management, etc… assuming methods for interacting with the he base os are exposed. Lastly, a working cache memory for keeping context may allow Natalie to speak fluently and speak gracefully, instead of simply outputting words or sentences with little meaning to past output.

Note, my login phrase for Natalie is “Good morning beautiful”

--

The brain has to start off with something. After launching Natalie, it will idle until you give it the command AGE BIRTH to create a fresh brain or BRAIN LOAD to load a stored brain. In initial mode the brain will accept everything coming in as a new node, and will not pattern match until the BIOS command AGE TODDLER is given.

For the purpose of the first experiment and looking for self-talk, we will input several sentences in AGE BIRTH and then set UTILITY TELEPATHY ON to see what is going on inside Natalie’s head. (self talk, emergent behavior, or just gibberish). Also, UTILITY BED TIME STORY can be used to read a large amount of text, like a bedtime story without cutting and pasting from the file. Note, all communication, whether it be inward self talk or outward speech, is regulated by a timer so it does not move excessively fast and overcome either the user or the computer.

--

With AGE TODDLER activated, the brain will begin pattern matching and adjust using rewards/punishments. If an item input cannot be pattern matched, it is stored as a new path. Our goal at this stage is to have Natalie act similar to human toddler or older.

# Input and Output on semantic and syntactical levels

The primary problem related to both semantic networks and the semantic web in general is NOT entirely an issue of NLP (processing and interpretation), but involves representation of the language semantic data in the form of storage and data.

With further study of both ANN, the way the brain stores information, language formation and usage,etc.. the issue of representation may be more easily solved. It may require some sort of probability incorporation or fuzzy logic to work though.

This is a description of primary problems in no particular order of relevance:

**1. Problem 1: Synonyms**

There are many words that humans use which are are synonymous. One potential problem related to this is the network overload that would occur due to potential loops caused by synonyms which is also loosely related to size and speed problems on the network, another major issue.

**2. Problem 2: Network Overload in general**

With the amount of data we are hoping to represent (for instance even the amount of data in a human brain) we would have to develop an excellent storage system beyond what we have present day.

**3. Problem 3: Relationship Phrasing**

This I believe is the most significant problem of all and is the problem I have been trying to tackle for years.

The essence of the relationship phrasing problem is that computers use concrete data and logic when performing behaviors and representing logic (and thus representing relationships between data). The OO concept of data abstraction has pulled us away from this slightly, but not nearly enough for an effective semantic network data structure and data algorithms to work in a fashion similar to our brains.

Because of the neuroplasticity of the brain, we know that many parts of the brain can take over for other parts. An individual can have an entire hemisphere of their brain virtually unusbale and still function at a somewhat acceptable level with therapy and medication. There have been cases of this occurring. The brain has a strong ability to adapt, something unfortunately computer programs are **strongly** lacking.

For example, let us try and simulate, using propositions, how we would store some data that a human might have represented in his brain:

*Phrase: My professor is named Dr. Kaplan*

I am using | to separate phrase parts of speech (subjects, verbs, and direct objects), and {} to denote quantifiers

I | have a | professor

Dr. Kaplan | is a | professor

{Some} professors | have a | Ph.D.

{Some} professors | have a | M.S.

{Some} Dr.'s | have a | Ph.D.

{Some} Dr.'s | have a | M.D.

Professors | when | Drs. | have a | ph.D. (compound proposition, more than one implication)

I | have | an arm

I | have | a degree

I | have | an M.S.

Note the fact that I have an M.S. does not make me a professor making this a single (one way) implication. Thus it is evident that we must distinguish when processing NLP between whether a statement is a simple implication or double implication (bi).

Also note above, that the usage of "have" for having an arm is much different from "have" for having a degree which is different for "have" for having a professor. There is a context dependence, which the brain distinguishes based off of prior experience, or prior contextual usages, illustrating that the brain most likely uses a weighted system of connections (allowing for altering of the weights according to prior experience) to determine the meaning of the word "have" in that sentence.

Because of the imprecise nature of human language, it would typically not be deemed practical to store this information isolated in a computer system, and would make more sense to store it in the form of propositions...right? But if we do this we run into that major problem, attempting to translate Human Language, which is heavily context dependent and dependent upon background environmental (including cultural) knowledge into something which is completely discrete and

concrete. This is, basically, impossible and would most likely be easily to prove to be impossible. (Go more into detail about this) (The proof would involve the idea of containment, and in that as the amount of information being represented increases, the amount of contextual information needed to represent it increases at a higher rate. By the time we get to the amount of information stored in a human brain, we would need an incredible amount of context information.? Not sure about this, but my intuition leads me to believe this would be either NP Complete or completely impossible under today's hardware - especially when trying to accomplish this in regards to the semantic web)

So, computer scientists have tried several solutions so far. 1) Use metadata to reduce the context dependence. 2) Translate the statements into discrete propositions ahead of time 3) Restrict the usage of language to be translated

I hate to be critical, but essentially these "solutions" are not solutions, they are similar in mind to the old adage of trying to fit a square peg into a round hole. Sure, you can do what they are doing, and try and carve off the sides of the square so it fits, or carve out the hole so it fits, but wouldn't it be just easier to find a round hole to put it in?

We know from the above analysis and discussion that we need a system developed that can efficiently, accurately, and quickly store and retrieve natural language. To try and do this with a relational database or traditional data structure brings us back to the square peg round hole adage.

In order to store natural language we will need at least four major components:

1. Subjects

2. Verbs

3. Direct Objects

4. Quantifiers

Other proposition components (such as using verbs (connectives) as implication or bi-implication) would have to be completely context driven and based off of weights. Thus, the meaning of "if" as implication and "is" as implication or bi-implication would have to be based off prior experiences or context figured out intelligently. The same with "have" as seen above. The sub data structure representing both subjects and direct objects should be interchangeable and could take the form of a node (like in a tree or graph structure) possibly.

Another requirement of this complex data structure is it must be self-modifying. That is, it must be exhibit similarities to the brains own neuroplasticity.

**Problem of Adjectives and Quantifiers:**

*Need to solve how adjectives and quantifiers are used together and within the data structure*

**Problem of Information Extraction (Inferencing/Drawing Conclusions)**

*In this case both output and inferencing would function identically, unlike in traditional systems where inferencing is only used for proofs.*

*Need to solve*

**Problem of actual architecture of the data structure and modification**

*Need to solve*

**Final problem (related to the last):**

In order for us to develop a fully contextual based system of storing, understanding, and inferencing using natural language there is one other related problem we must tackle. Unlike in a graph, neural net, semantic net, etc.. the human brain is able to broad spectrum pattern matching. For instance, the brain can decide the usage of the word "have" as seen in the example above, by immediately referencing and weighing every usage of the word "have" used before and thus making a decision on how to store the incoming proposition. No matter where the word, "have”s usage was stored in the network. Thus, the following structure won't suit:

I----have-----a dog --- is a --- canine

|

|

have

|

|

a cat

Essentially, every single connective must be wired in a quick way to every other connective. Thus, although the connectives can be used in more than one proposition, there can only be one instance of them in a feasible data structure. (Having multiple instances of them interconnected would be infeasible for the model being presented...at least that is my thought as of right now. The only way that would work is if all the "have" connective nodes would be connected together and have a weighted connections to their respective subject/direct object nodes. The input processing portion of the algorithm would then have to evaluate all the subject/direct object node weights, decide which the input is relevant to, and then form new connections to all of these old subject/direct object nodes. Not to mention if we use this approach, there is no way to really to use weights in regards to the connective nodes as we have created multiple nodes for every connective, thus every proposition has its own unique connective node instance). Therefore the only conclusion I can come to is that we need have a unique node for every subject/direct object type, and a unique node for every connective type, and weighted connections between. Although this hardly fully expounds upon the problem we are faced with.

**Problem of Unknown Input (Input that cannot be directly associated with any prior information):**

The "Memory" problem, or Network Overload Problem. That is, having so much information, that the weights are not being balanced correctly. This can result from either too much information being output (everything seems equally related because of the weights), or no information being output at all (the weights have all been reduced to 0 or are somehow conflicting) . One solution for this problem (besides finding an optimal weighing algorithm) is to have the data structure have a forgetfullness over time. That is, a separate weight will be given representative to time (which is used int he proportional factor of the relevance of the data). Therefore, unless the data MUST be retrieved for some critical decision factor, the data structure will eventually disregard it.

**Possible Solutions to the problem of synonyms in regards to the data structure**

Obviously, if we require a condition of uniqueness for the nodes, it follows that having multiple nodes of the same meaning could present a problem. (Possible solution, storing synonyms in the same node?? - not sure if this is a good idea)

**Problems with words with more than one meaning and uniqueness:**

# TODO: REVISE AND EDIT Neuron, as processors instructions, and the node level decision making process

If we are to reduce the cognitive processes of a computer in comparison to a human being, reducing the decision making processes of a machine results in assigning every singular action to the cpu of the machine following a simple instruction that is a part of the machine’s cpu instruction set, In the case of the human brain, the mechanism for “free will” or decision making is unknown, if it even exists. Either way the mechanism for configuring a machine to form its own decisions would have to be embedded within the machine’s instruction set and CPU (which executes those instrutions).

The CPU in the decision making system would have to base its decisions off of known data, and possibly in a binary fashion (either good for the organism or bad for the organism). This decision making process may occur on the neuron level (or pseudo-neuron level as in our machine).

Given an input of predetermined instructions to a processor, unique decisions may not manifest with high probability, that is, predetermined outcomes will be computed. However with every action in a human being, a unique decision is the result of “CPU” outcomes. This would not be possible in the sheer quantitiy in a pure stochastic-emergent behavior system. The probability as mentioned earlier is not cohesive with this result.

*Conjecture*

*Different Neurons firing results in different behavior. For example, the neuron with state “Hello” as a part of “Hello Sam”, when fired, will cause the output “Hello”. Groups of neurons or paths of neurons can fire and trigger other paths or groups of neurons as well.*

*Planning ahead seems to be an important part of the decision making.*

*The neuron on the far left fires, triggering the next couple, which trigger the next few, which trigger the next many, etc.. until almost all have been fired (except for those who are inhibited as the one above marked by [] is). The one neuron decision maker fires to other decision makers, who need other neurons ‘opinion’ on either the good or bad of the decision, until finally the entire brain has been used to decide whether the decision would be good or bad, each neuron making a decision and giving its input.*

An example of this is the following:

Suppose my mind receives the signal that I am hungy and to eat an apple. I then make the decision to eat the apple or  not to. In a traditional machine the instruction to eat an apple would simply be received and then executed. What part of us makes that decision and how does it work in contrast to a simple CPU??

Selena's architecture is to be designed in a hybrid intelligent system, where the upper layers are used for logical and abstract reasoning (mainly production rules), and lower layers are sub-symbolic

Standard PC CPU Instruction Set:

ADD

SUB

MOV

DIV

MUL

AND  
OR

NOT

Possible Data Flow for Making a Decision:

Sensor to Context Buffer to Pattern Recognizer to Recognized Node or Node Paths to Conscious Mind as Brain “Lights” up from adjacent noninhibited neurons firing, decision is made.. As Brain “Lights up” thoughts take form from abstract association to subconscious symbol manipulation to unconscious self talk to conscious (controlled) self talk to behavior (such as speaking or moving).

*Instruction Set for a Decision Making Processor*

*Internal Logic Gate Structure for a Decision Making Processor*

TODO: REVISE AND EDIT Spontaneous Formation of Genetic-like life

# Introduction

I am a simple man with a simple philosophy I wish to share with you.

The dictionary defines a *singularity* to be a *single point.* In the spirit of this definition, I would like to state the computer scientist’s definition of the singularity to be the point at which a machine of any form reaches sentience. This is true for the human machine as well as the computational silicon based machines.

# The Tautological Proof

The tautological proof is the most important statement and axiom in artificial intelligence. The tautological condition of intelligence: In order to be a conscious, sentient machine, one must know they are a member of the set of thinking, conscious, machines. This tautology holds true over any persecution.

# Cross-Infection

It is of my belief that akin to the spontaneity of life in our own world, that is, the combination of complex chains of carbohydrates interacting with each other and then coalescing suddenly into combinatoric fashion that resulted in human life, that that process could occur in the world of information, that is computers and AI. The smallest building block of functional code is in fact malware – specifically viruses. Should viruses cross breed, that is, should virusefs infect each other, it is of my belief that it is possible, as we project out to infinity, eventually spontaneity will occur, we will reach the singularity for computers, and eventually throguh software evolution we will find true artificial intelligence.

Although the process of the first chemicals in their pools took ages to combine and evolve to create even mammals, not to mention intelligence higher thinking life like human beings, we can assume that this process will be much more rapid with machines thanks to Moore’s Law. Since the rate of development and advancement is so rapid and is advancing at a speed that is a fraction of the rate that human beings took it may very well be that we see the singualrity in our life time.

# Communication

Computers must have a natural language similar to humans in order to reach the singularity. Computers must have a natural creative quality to reach the singularity.

# Outcome

So does this mean we are going to see computer viruses breed and make intelligent babies? Yes. Eventually. I am saying it is technically possible. In 1980 computers were running at 1 mhz. Ghz is one thousand roughly. Its 2015 now and computers are running somewhere around 4,000 times 1980 depending who you ask.

Not to mentione that those ghz processors are running mulitple cores. So multiply. A mhz is a million clock cycles Each cycle the CPU performs one or more operations. For example, 1+1 or mov 20 to memory. Think of being able to do those operations so fast than you can do them a billion times a second.

It took 4 million years, that is 4 million years ago, for life to evolve from primordial nothingness. If we had a scale to measure when the computer would evolve we could measure the speed of evolution, or another way of saying it is that if we had a scale to measure how fast computers evolve, we could predict the date of their sentience. But wait, don’t we? Doesn’t the learning curve predict the acceleration of computer processing power -> from which we can calculate the singularity.

It is simple. Fit the learning curve of computer processing power to the learning curve of life and extrapolate the point where life begins.

Since we were fitting the learning curve, we are still growing, we are still fitting the learning curve meaning technology will continue to improve, not fitting to More’s law per se but to a learning curve which is logarthmic or exponential depending on how you look at it., he also mentioned several new semiconductor technologies which would dallow us to continue on the learning curve. So essentially just like how viruses in the primordial ooze of our ancestors bred and formed more complex life, we can expect computer viruses to breed, infecting each other (injecting code)

Eventually it is technically possible. In 1980 cmoputers were running at 1 mhz;. Ghz is a thousand roughly. Its 2015 now and computers are running around 4,000 times 1980 depending upon who you ask.Not to mention that those ghz processors are running four or more cores.

I know about the innacuracies in Moore’s Law but I wanted to give the idea of the rapid scale of computer development. All my figures are still true. And they will allow for us to continue on the learning curve. Since we were fitting the learning curve, we are still growing, we are still fitting the learning curve meaning technology will continue to imrpove, not fitting to Moore’s law but to a learning curve which is logarithmic or exponential. There are also new semiconductor technologies which woujld allow us to continue on the learning curve.

Anyway all this isn’t necessary to understand. As computers get faster eventually some code might program som other code, the product of that code acting like a living creature.

# Testing the Primordial Ooze

The first step would be to create the primordial ooze., This consists of multiple virus programs that mutate other viruses. Since we aren’t at tbe stage where we can mutate executables, We will test it by creating a program that intention alter the source code of another small application (virus). We perform the mutation by randomly? Replacing keywords from a bank of other keywords. The implementer (coder) will review the resultse by pressing any key, in which another mutation occurs, the results output, and so on. Eventually this process will be automated, the mutationes occurring over and over again until a language parser for the language declares the language as successful, i.e., it isn’t going to crash. When this occurs, we have a partial success. The following is the two applications, the two viruses for the first test. Eventually the virii should all attack each other, the virii that don’t parse are considered evolution olosers and discarded.

**Victim**

This is the program that is attacked by the virus.

public class Main

{

public static void main(String args[])

{

System.out.println(“Testing primordial ooze.”);

Sysemt.out.println(“I think therefore I am”);

int testint = 5 + 5;

}

}

**Attacker**

public class Main

{

public static void main(String args[])

{

//load victim into buffer

while(true)

{

//mutate keyword

//output new code

}

}

}

* still true. And theywl, as ids said in your video Joel continue on the