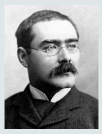
Thinking Yes and No

"Introducing the AxonP Cognitive Architecture"

"I keep six honest serving-men, they taught me all I knew;

Their names are What and Why and When,

and How and Where and Who..." *- Rudyard Kipling (1902)*



Introduction

*This article iintroduces **AxonP** - a Cognitive Architecture and working model of the brain. A Cognitive Architecture tries to explain how the brain works. A myraid of such architectures abound, ranging from symbolic to connectionistic. A review [1] done in 2018 estimated there were already 300 such architectures.*

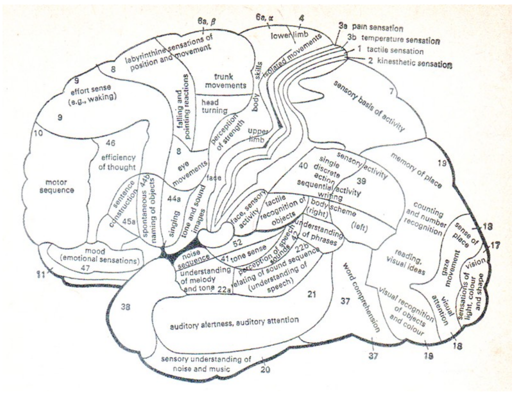
AxonP is a holistic, top-down, and biologically-inspired architecture, not basing on any existing ones. It is built upon good old-fashioned AI (GOFAI, a term for traditional high-level symbolic processing).

*AxonP is short for **Axon Project**. It is named after the Axon Idea Processor, a Windows-based tool for visualising ideas developed by the author in the early 1990's. This article focuses on the assumptions, principles, and design, while the details are best gleaned from the working model in the given link [5].*

1. Modelling the Brain

A Cognitive Architecture models the brain functions. A model is a representation of the essential features. The brain has regions with specific functions, such as show in Fig. 1.

Fig. 1. *Brain map by Karl Kleist, 1934*



A complex model can be divided into simpler, interlinked sub-models working in unison. The more sub-models are engaged, the better the understanding of the situation.

A sub-model in AxonP is called a **Cluster.** A Cluster is liken to a Class in Object-Oriented terminology, and it contains data and algorithms. AxonP has over 50 Clusters. Some typical ones are:

Thing – This Cluster is about man-made, countable, tangible, non-living things.
Scene – A scene binds location, events, people and objects.
Relation – This is a collection of links between Clusters.
Word – This is a central repository of all words. Only root words are considered. This Cluster corresponds to the Wernicke's Area.
Action – This Cluster handles perception and generation of physical and mental actions.
Self – Self takes ownership of events and things, and it is the soul of the machine. Modelling the Self is a first step towards consciousness. However Self is a mere simulation and not meaningful because Self is easily cloned.

2. Cluster Organisation

Clusters can be arranged in many ways, such as hierarchical (top-bottom), lateral (left-right), front-back, by types, by processing speed, etc.

A top-down hierarchy ranges from abstract at the top, to concrete at the bottom. At the top level, Clusters can be arranged by specialised **Domains**, such as Spatial, Entity, Social, Visceral, etc. At the bottom level, Clusters can be arranged by input/output **Channels**, such as Eyes, Nose, Tongue, etc. [Fig. 2]. Clusters in one hierarchy tends to cross-link with those of a hierarchy with similar abstraction and speed, otherwise the communication is out-of-sync.

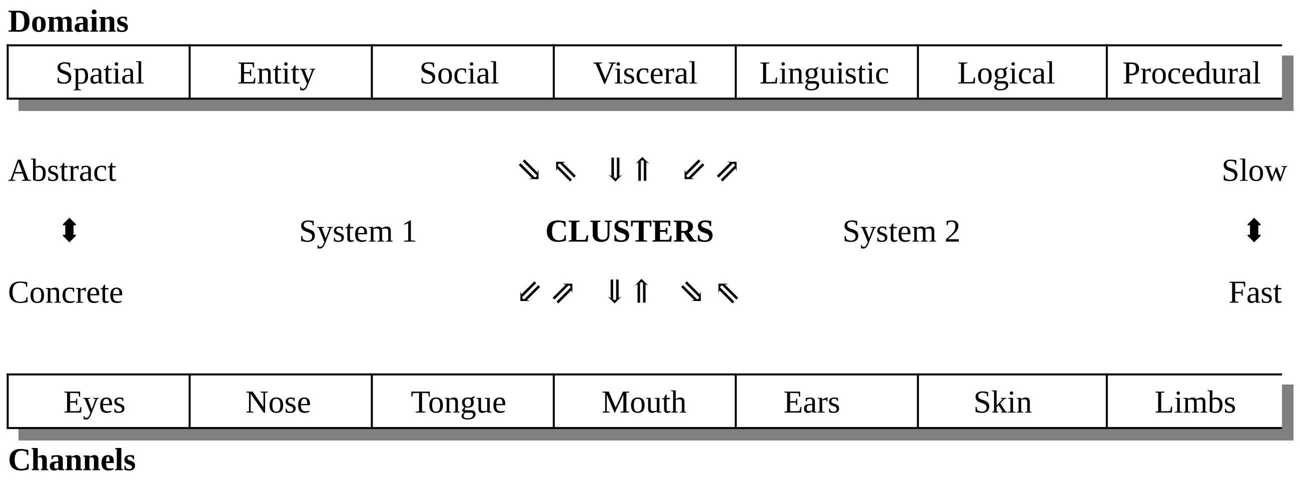


Fig. 2. *Clusters can be arranged by abstraction levels, Domains and Channels, and speed.*

System 1	System 2
Right-brain	Left-brain
(Big) Data	(Big) Algorithms
Space	Time
Declarative	Procedural
Implicit	Explicit
Visual	Verbal
Parallel/Fast	Serial/Slow
Analog	Digital
Intuitive	Analytical
Form/Noun	Function/Verb

Table 1. *System1/System2 characteristics.*

Clusters can also be divided into **System1/System2**, as described by Daniel Kahneman in his book *Thinking Fast and Slow,* [2].

Basically, S1 characteristics are the result of neurons firing when a input threshold is reached (regardless of sequence), and S2 characteristics are the result of input occuring in some sequence. Every Cluster contains a mix of S1 and S2 in various proportions.

3. Linking Clusters

In the human brain, neurons (gray-matter) are nodes, and axons (white-matter) are links. A node is linked to many other nodes. A link between node A and node B can be written as A ⇒ B, and logically equivalent to (*not* A *or* B). The proof of this is by listing all the possible cases in a Truth Table:

A	B		A ⇒ B	*not* A	*not* A *or* B
Yes	Yes		Yes	No	Yes
Yes	No		No	No	No
No	Yes		Yes	Yes	Yes
No	No		Yes	Yes	Yes

Table 2. *Truth Table to prove that (A ⇒ B) is equivalent to (notA or B).*

The above equivalence implies that the billions of links (⇒) can be **entirely** replaced by logical *not* and *or*. Hence logic and negation are fundamental to thinking. Essentially we are thinking Yes and No. To understand something is to know what it is not. Peter's Law #21 [3] says that a true expert knows exactly what cannot be done. Proof by contradiction (*Reductio Ad Absurdum*) is the basis of **Prolog**, a programming language for logic.

While individual links are too small/many to show up in a brain diagram, **Pathways** are major brain connections visible to the naked eye. Horizontal pathways link Clusters with similar abstraction levels, dimensions and speed. E.g. The Corpus Callosum joining the left and right brain hemisphere. Vertical pathways join Clusters at adjacent levels of abstraction, e.g. the Arcuate Fasciculus joining the Wernick's and Brocas region, also the Dorsal and Ventral visual pathways.

Observable ⇧	**Controllable ⇩**
Reality	Expectation
Convergent	Divergent
Feedback	Feedforward
Sensory/Perception	Motor/Inference
Seeing/Reading	Drawing/Writing
Encoding	Decoding

Table 3. *Vertical pathways can be Observable* (⇧upwards) *or Controllable* (⇩ downwards). *This table highlights the characteristics of both types.*

Control Theory says what is Observable is Controllable, and conversely what is Controllable is Observable. We can express this equivalence as ⇧⇔⇩. The following Fig.3 illustrates the equivalence:

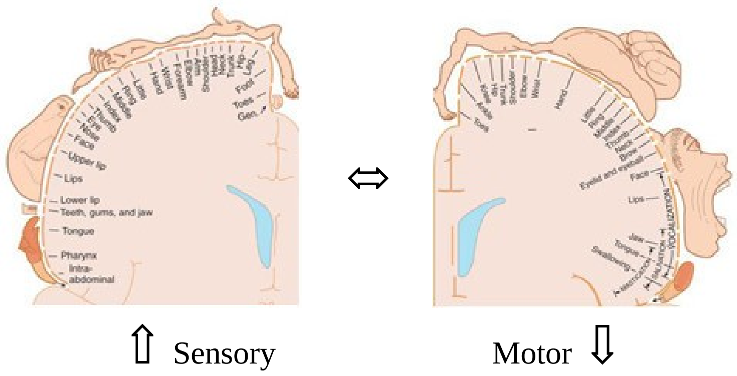


Fig. 3. *The Sensory Homuculus (left fig.) and Motor Homunculus (right fig.) showing the sensory-motor (observable-controllable) mirror image.*

4. Diagramming Method

A Cluster has 3 logical types:

1. **Positive Cluster** represents known information.
2. **Negative Cluster** represents unknowns, goals, intentions, gateways.
3. **Embedded Cluster** interfaces with the body, and is a combination of both above.

Negative Cluster	Problem	Question	Unknown	No
Positive Cluster	Solution	Answer	Fact	Yes

Table 4*. Thinking involves problems and solutions, questions and answers, unknowns and facts, and ultimately yes and no.*

To represent Clusters and Links, we will use a simplified but extended form of **Ferguson Diagram** (named after R. J. Ferguson, 1977). [Ferguson Diagrams are not well known and no reference is available]. We will assign the shapes ◒ ◓ to positive and negative Clusters, and a circle ● to Embedded Cluster.

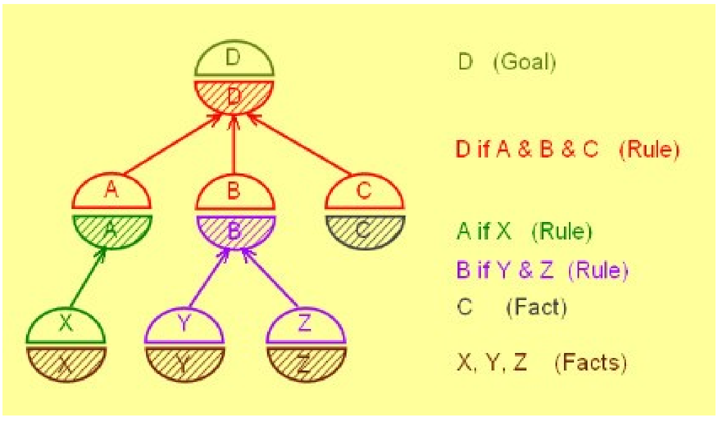


Fig. 4. *A simple Ferguson diagram consisting of semi-circles, shown with its logical equivalence shown on the right. A matched pair forms a full circle. AxonP applies this diagramming convention to represent entire clusters.*

Positive ◒ and negative ◓ entities are logically and visually mirror-images of one another. A positive entity (e.g. a statement) usually has the (+) omitted by default. A negative entity (e.g. a question) has a (-) in front, or equivalently a (?) at the end, or some other notation. Hence a statement can be inverted to become a question by simply adding a (?) at the end. For example "The sky is blue" (Statement) becomes "The sky is blue?" (Question).

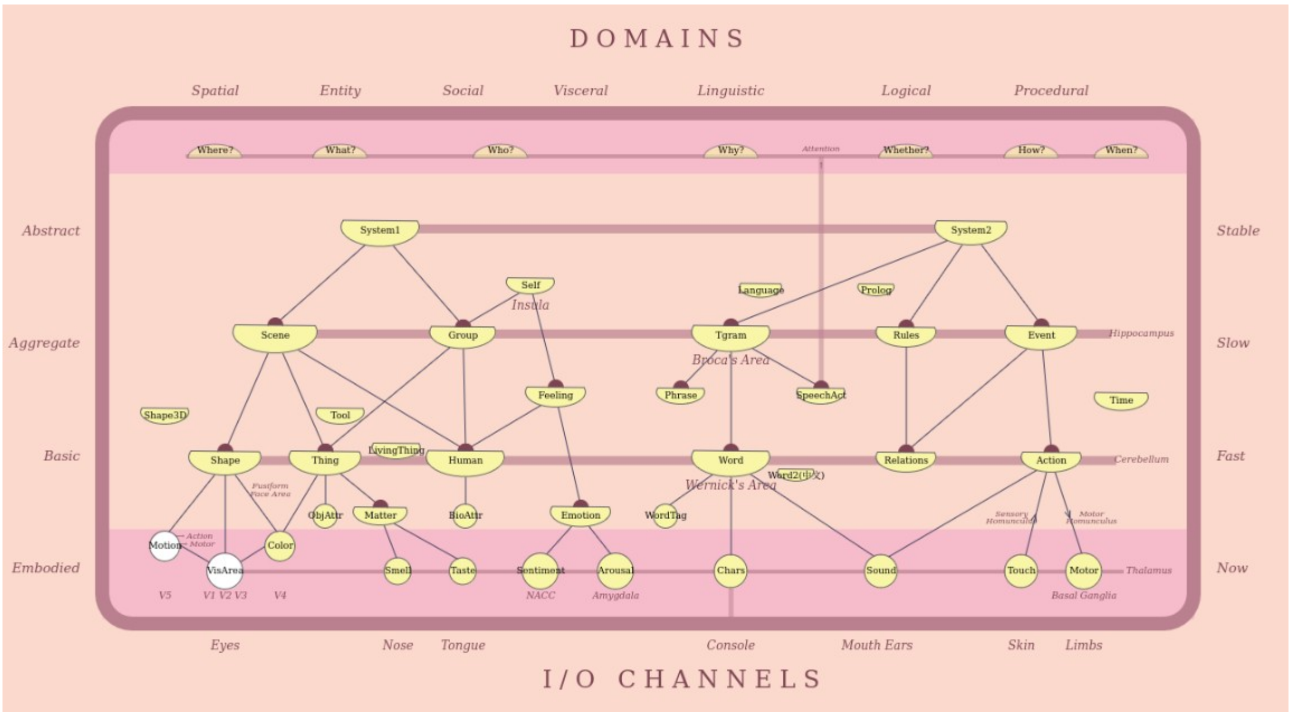


Fig. 5. *The AxonP Cluster Diagram with minimal details. Note that the Right-brain is drawn on the left of the page. This assumes that "right" is with respect to the person facing you.*

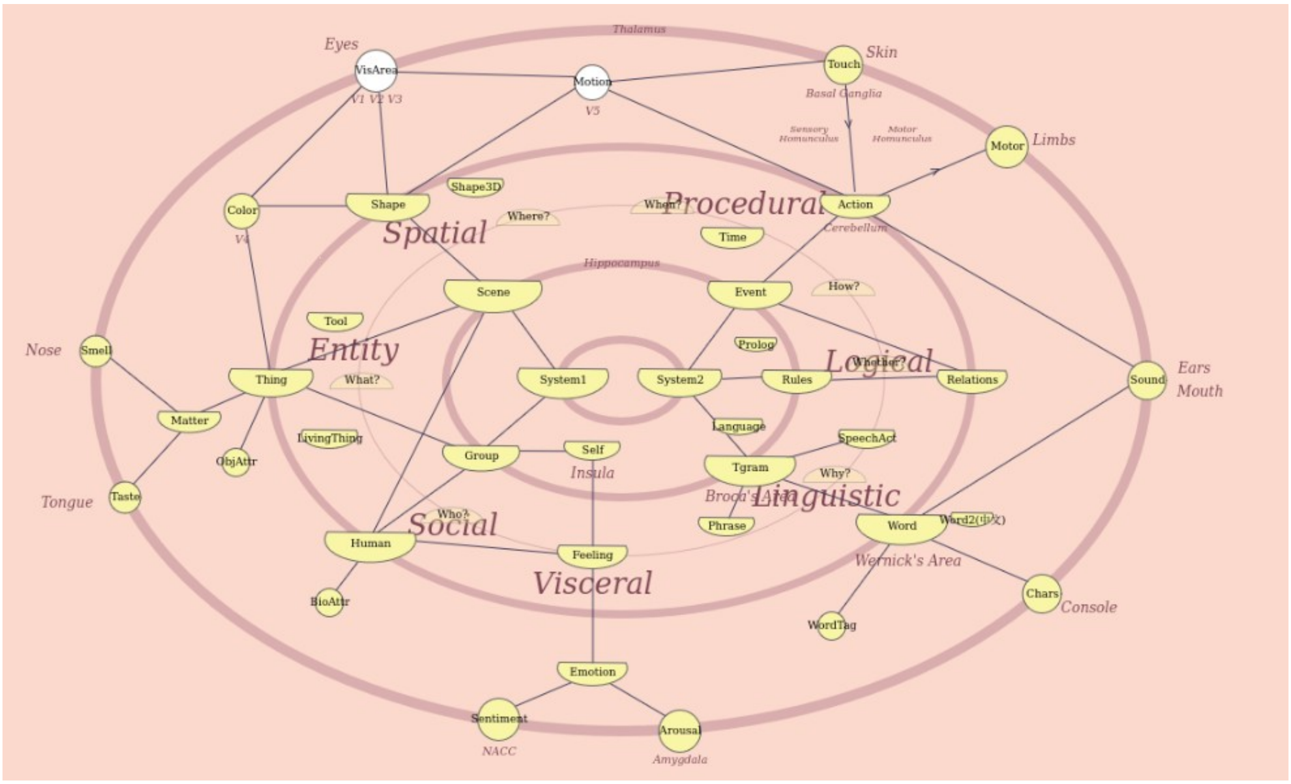


Fig. 6. *AxonP Cluster Diagram displayed radially. Note the Motion Cluster is now wrap-around.*

5. Types of Links

Links are the attributes (or slots) associated with each Node. For example, for a human, the attributes are age, and gender. Links are also known as **Relations**. In 'Einstein likes music', the Relation is 'like', linking the nodes 'Einstein' and 'music'. Examples of Relations are: like, near, own, is-at.

There are 4 types of Relations:

1. Basic Relations are extracted from the Nodes in each Cluster.
2. Symmetric Relations are bi-directional, e.g. A is near to B implies B is also near to A.
3. Transitive Relations are uni-directional, e.g. A is bigger-than B, and B is bigger-than C implies A is bigger-than C.
4. Causal Relations e.g. 'because-of', are used for answering Why questions. Note that Correlation Is Not A Cause (CINAC).

A Relation may belong to more than one category, e.g. A Causal Relation may also be Transitive.

6. Modelling Language

English is the chosen first language. Input-output is in short and simple English, and no effort is made to produce grammatical and formal prose.

Words are abstsracted into root form. **Root Words** do not have tense (past, present, future), and singular. Root words are in lowercase unless they are proper nouns. (e.g. not monday but Monday, whereas both apple and Apple(™) are valid). **Functional Words** have no content.

Words are classified by **WordTag**, a more elaborate kind of Part-of-speech. Examples of WordTags are ANIMAL, APPEARANCE, BODY, BUILDING... There are over a hundred tags in AxonP. Words that have multiple WordTags are called **Ambiguous Words**. These require special handling.

A **Phrase** is a hyphenated set of words (e.g. part-of, as-soon-as), and it is considered a single Word. This assumption greatly expand the scope and power of words. (e.g. *sibling* becomes the phrase *child-of-the-same-parent*). The gist of an utterance is processed in the **Telegram** Cluster.

AxonP also has a second language (Chinese) and can perform very basic translation. A Chinese translation of this article is at *http://axonp.com/axon-ch.pdf*

7. Algorithms

Big Data requires complementary Big Algorithms, i.e. the functions in every Cluster. While Big Data can be automatically acquired, algorithms require much hand-crafting. Examples of Cluster-functions are:

- *Language.parse()* - Analyse the input text. In this and the following examples, the function name (i.e. parse()) is preceded by the cluster name (i.e. Language).
- *Logic.parse()* - Analyses a Prolog statement or query and change it into a logical form for further processing.
- *Thing.describe()* - Describes the specified Thing in simple English. Many other Clusters also have this function, e.g. *Human.describe()*.
- *SpeechAct.command()* - A SpeechAct is a kind of Action for classifying an Input as a Question, Statement, Command, Greeting, etc. Based on the SpeechAct, an appropriate response can be issued e.g. a Question gets answered, a Statement is remembered, a Command is carried out.
- Emotion.tally() - Performs basic Emotion detection based on what is transpired in the recent past. The 4 basic Emotions tracked are: Happiness   Sadness   Anger   Fear   (**乐 悲 愤 怖**). Modelling emotions are mere simulations. No machine really fears being incinerated.

A major design consideration is to use Prolog, since Prolog's resolution methodology is very similar to how the brain works. AxonP implements a Prolog interpreter in Javascript [3]. Standard Prolog terminology is used, e.g. Atom, Term, Functor, Argument, Variable, Fact, Rule, Goal. Variables are the unknowns, and variable names in Prolog are capitalised. AxonP requires, in addition, that Variable names are entirely in capitals so that Proper Nouns need not be Variables. Hence John is a constant, whereas JOHN is a variable. As Prolog is built-in, you can also input Prolog codes directly into AxonP. To test the correct functioning of the Prolog interpreter, a 21-rule Einstein Puzzle [4] can be loaded and solved .

8. Implementation and Testing

AxonP is implemented in vanilla Javascript, a versatile language with very good graphics. AxonP caters to developers by providing details down to individual functions and trace-logs. The effort required to develop AxonP (or any complex system) is mostly in the design phase rather than programming. An early implementation of AxonP consisted of:

- 47 Clusters
- 121 Rules
- 57 Relations
- 95 WordTags

Being a top-down design, these numbers can be easily scaled up without significantly increasing hardware resources. You can access the current implementation from the AxonP website [5].

Tests (or Unit Tests) are built into each Cluster. A Test is simply a line of Input, which produces an expected response. Running all the Tests after each program change helps to verify the correctness of the change. A Test usually involves many Clusters. Certain Tests are meant to be carried out in a specific sequence. For example asking “Who am I” will not get a correct response unless you previously made a statement as to who you are. During a (Test) session, working memory is cummulative, and is cleared when starting a new session.

Here are examples of some Tests and its results, including tests on question-answering, commands, and logic:

- *Who are you? - I am Self*
- *Where are you?* - Self is-at Computer
- *What is the color of green?* - ● <green colored dot>
- *What is the shape of a penguin?* - <the penguin icon>
- *List animals* - ANIMAL albatross alligator anteater...<over 130 animals>
- *Translate human to Chinese* - 人类
- *世界你好* - Currently cannot handle Chinese input
- *Is speaker on?* - Speaker is on
- *Turn-off the speaker* - Speaker is now turn-off
- *Tim has-mother Anna* - Added Relation: has-mother(Tim,Anna)
- *Anna has-father John* - Added Relation: has-father(Anna,John)
- *has-ancestor(Tim,John)?* - Yes
- *has-ancestor(John,Tim)?* - No result
- *Load puzzle* - Loaded 21 rules from Prolog.PUZZLE

- *solution(WATERDRINKER, ZEBRAOWNER)?* - solution(Norwegian, Japanese)
- *Does dragon exist?* - Yes. is(dragon,imaginary)

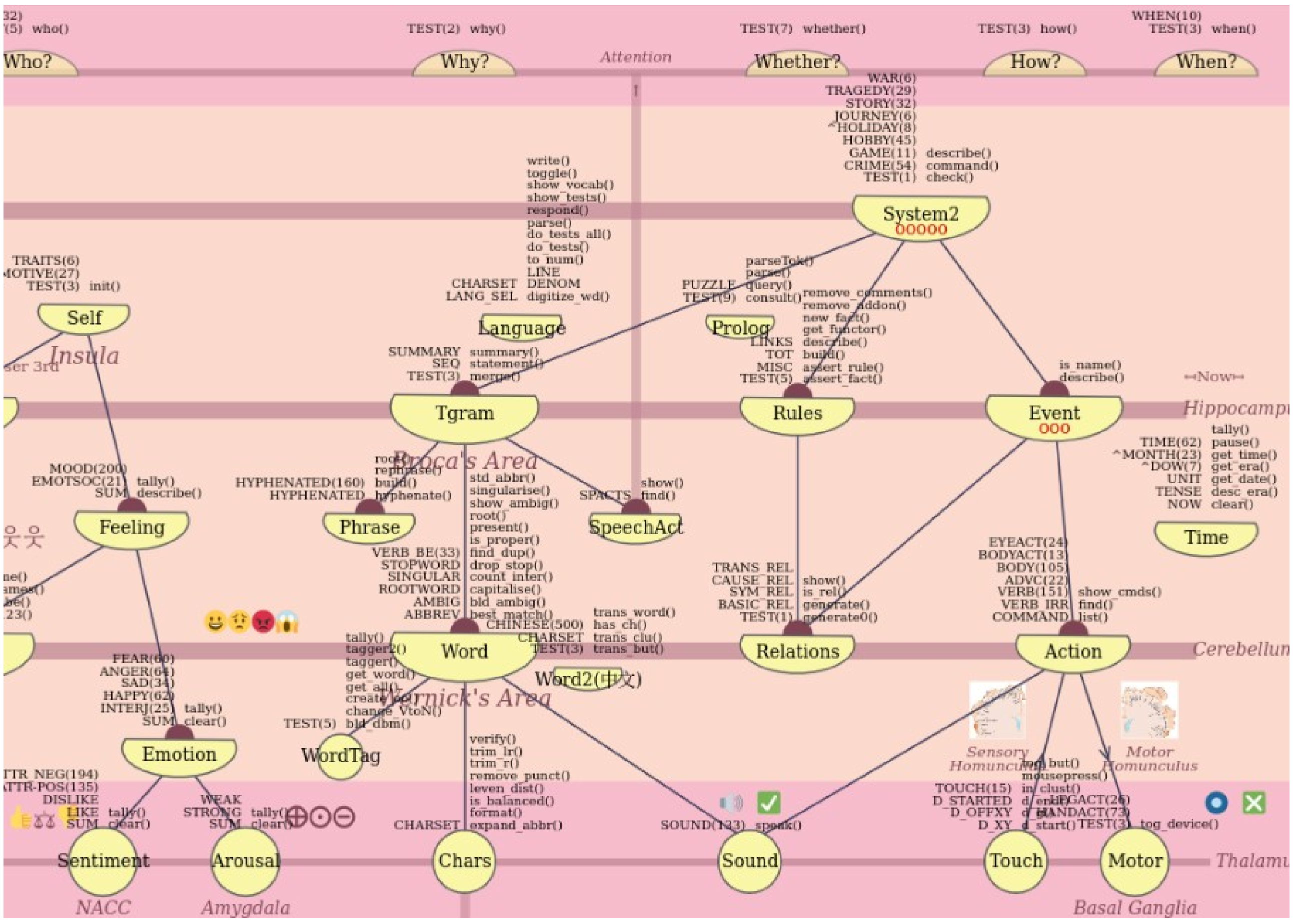


Fig. 7. *Progressively more detail is shown when you zoom in. The text columns above each Cluster (yellow) are the Data (on the left and capitalised) and Functions (on the right, with ending brackets). The o (in red) indicates a Node. During operation, the yellow clusters turn to white to show there is activity in the cluster.*

Conclusion

AxonP provides an efficient framework for complex applications like autonomous systems, chatbots, and design/implementation of major constructions projects, etc. AxonP's design sets the right direction for future enhancements, and helps to push back the wall of complexity.

Here is a list of additional features/functions to consider:

- Asking questions and offering suggestions when appropriate.
- Detecting/modelling intention and long-term goals.
- Decision making based on emotion and value system.
- Incorporating Fuzzy Logic and Fuzzy Prolog

- Integrate with neural networks.
- Enhanced diagramming in 2½D or 3D.
- Parallel or pseudo-parallel (time-sliced) operations.
- Looking-up external information sources.

“I know a person small - She keeps ten million serving-men, ...

One million Hows, two million Wheres, and seven million Whys!” *- Rudyard Kipling (1902)*

References

[1] Kotseruba, I., Tsotsos, J.K. 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artif Intell Rev* **53**, 17–94 (2020). https://doi.org/10.1007/s10462-018-9646-y

[2] Daniel Kahneman, 2011, *Thinking Fast and Slow*, Macmillan, New York.

[3] Peter H. Diamandis, *Peter’s Laws:* https://www.diamandis.com/laws

[4] *Solving Riddles with Prolog and ES6 Generators*, https://curiosity-driven.org/prolog-interpreter.

[5] AxonP Website: https://axonp.com/axon.html

About the Author*: Chan Bok started his career in the Port of Singapore in the early 1970s as an R&D Officer. Since then he worked in IT and consulting. Winning the top prize in the First National Software Competition was one of his achievements. He is also the author of the Axon Idea Processor, a tool for working with ideas. Mr Chan has a Honours Degree in Mathematics and a Masters Degree in Industrial Engineering, both from the University of Singapore. AxonP is an on-going project and he would appreciate any feedback or support. He can be contacted at:* chan.axon2000@gmail.com