# Multilayered Model of Speech

Andrey Chistyakov*

**Abstract**

Today there are no grammar systems, which allow the creation of a fundamentally new word and concept. All existing grammar systems only work by referring to previously chosen terms, on the bases of which all definitions are created. Implementation of operation on grammar systems are introduced for creation of new terms. The main prerequisite of the research was the rejection of finding a universal solution, which is true for any person. Instead of this, all people were divided into groups according MBTI classification. Each group was expected to have standard perception of new knowledge. Assessment has been conducted to each group. A mathematical model was created as result of the communication. The scheme of dividing words and sentences into components are shown in the first section of the article. The second section shows a notation construction from the components and the notation packing in memory. The third section shows the ability for conscious memory access (self-action and self-image). As a result, the model of human speech was structured, in where it is possible to create new terms from new knowledge independently.

*Keywords*: Artificial Intelligence; Natural language processing; Speech recognition

## 1 Introduction

The problem of human speech modeling has been solved more than once, but each time it faced paradoxes and contradictions. The scientific approach was initiated by Boolean algebra, which transgressed into logic of predicates. The human speech is more complicated than the predicate logic, but at the same time it is free of paradoxes, which occur at the first or higher orders. This article is an attempt to solve certain problems by introducing special restrictions.

Speech analysis will be done in some consecutive stages. We should create automata, which can work over context–sensitive grammer, at first. This grammer will be defined in section 2. This definition based on concept of

*Email: andrey.chistyakov@ghoort.com

mask, which is responsible to symbol from ingress aphabet. The automata's transition function devides on some parts, which has specific logic operations over its. The main difference with previous studies is the specific function $\Phi$, which can represent any class as data and some data as the class. The section 3 describe the automata's algorithm. There introduced the simple context as the set of symbols between two terminals.

We describe the hierarchical system of one-modifying automatas in section 4. The automata system should recognise symbols in context-sensitive grammer. Analogies for the hierarchical system exists in scientific literature, but the article has the significant difference. The difference is in memory system with selfmodifications by the operations over the classes from section 3. The selfmodification ability has some special restrictions because we need to pass over infinite looping of any logical algorithm, related to Gödel theorem.

We now turn to generalized grammar in speech, generally independent of a specific language. Basic elements of speech are lexemes, they are the words with a meaning. We will use following lexemes - Nouns, Verbs and Adjectives. Nouns and Verbs are clear, because they exist in all human languages. Nouns names objects and subjects, Verbs - actions between them. Adjectives marks qualities and quantities. Lexemes do not exist by themselves, but only being bound to sentence with subjects, predicate, and other words, including function words.

Adjectives represent some qualities of the object, which a person initially receives from feeling an interaction with an object, so it is possible to say that adjectives are linked with feelings. Verbs represent actions, which are performed by a person or any other subject. Those actions are usually performed through muscle reflexes. Therefore Verbs are linked with muscle reflexes. Nouns serve as markers of feelings/reflexes triggers. It will be shown from which they originate. Basic sentence are some stable sequences of ingress and egress signals. In this sense, words or lexemes, as letters sequences, are also lower level sentences.

But speech consists of not only egress and ingress sentences, it also includes the human memory. Memory is presumed to be multilevel, objectual, and context oriented. It contains complex, processed reflexes. Memory is arranged in a hierarchical graph, its nodes are classes and objects. Class is a Noun collection of Adjectives and Verbs. Object is an implementation of class. Also classes can be represented as data and they may be included in the object. Based on this feature, classes can be converted between each other and packed into memory.

A concept of option is introduced for definition of memory work. This is the main innovation of the concept of the proposed model. In the option's scope there will be two possible options of information decryption. Both of these concepts are not equivalent and they can exist in two different people, but not together in the one person. Every option is fixed randomly, during the creation of the brain, and will not change during the whole lifetime of



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

the person.

A total of four independent options will be introduced, which means that a possibility of 16 different options of human speech are possible in this model. It is the most significant difference from all other attempts of creating human speech modeling mechanisms. It is presumed that this division was made possible in the result of evolution, for the interpretation of all incoming information, while every individual can intake only a part from the "raw" flow, to process and transfer to other individuals, prepared in way which can be used.

The possibility of introducing optionalization is grounded on different psychological theories, based on Karl Jung's archetypes [Jun71], primarily Myers–Briggs Type Indicator [MM95].

For the modeling of the brain I will use terms and principles from Computer Science, such as addressed memory, search operations of certain symbols in sequence and Boolean operations. Names for some terms are taken from Socionics, a special discipline about typology of the human personality.

## 2 Basic Definitions

Lets introduce basic definitions for the beginning.

**Definition 1.** Let a point be a pair from an address, where the address is a natural number, and value equal to 0 or 1.

**Definition 2.** Layer is one-dimensional finite array of points, with consistent and continuous numbering of addresses from 0 to $N$, where $N$ is the array length. The array length is fixed during the working process, but values of points are variable.

**Definition 3.** Let the stack be finite, consistent, and continuous array of layers.

**Definition 4.** Mask $\mu$ is a fixed, finite array of a pair $(i, [0, 1])$, where $i$ are natural numbers from 0 to $N$.

It is necessary to clarify that even though the mask is consistent, *i.e.* it has no points, which have equal addresses, but all addresses of the mask cannot be continuous. Mask's addresses should have gaps, and their meaning will be discussed below.

Basic operations, in the system of layers and masks, are offset by definition and by blocking the mask range. Suppose that at some point of time values, at a layer, are unchangeable. Then it becomes possible to check all layers, and find a combination of values in the layer, which are represented in the mask. The least address element, of first point of the layer, appropriates to the mask, is called **begin of block** or **offset**. The last address is called



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

as **end of block**. The gap between the beginning and the end - including, is called **block of layer**. A block of layers always matches to a specific mask. Operation of finding entry of a mask - is called obtaining an offset.

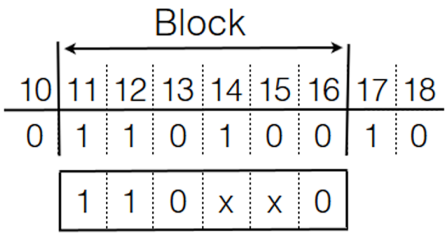


Figure 1: Mask

**Requirement 5.** Blocks on the same layer should not intersect each other.

**Definition 6.** Let $B$ be a block of a layer, which consists of serial, continuous set $(b_0, b_1, \ldots, b_m)$. Then it is possible to define the set of the Boolean function $\mathfrak{F}(B)$ of the type $f(b_i, \ldots, b_k) \to [0, 1]$. The set $\mathfrak{F}(B)$ will be closed under Boolean operations $\oplus$ (XOR) and & (AND). Let's call $\mathfrak{F}(B)$ as set of **Adjectives** and the result of these values as **quantities**.

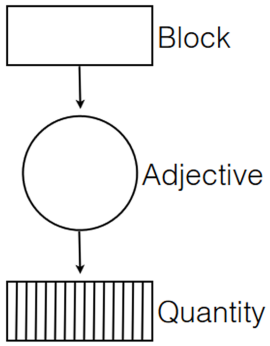


Figure 2: Quality and Adjective

**Definition 7.** Let there be a set of qualities $Q = (q_0, \ldots q_n)$ and block $B$. Then it is possible to define the set of the Boolean function $\mathfrak{P}(Q, B)$ of the type $f(q_0, \ldots q_n, b_i, \ldots, b_k) \to [0, 1]$. The set $\mathfrak{P}(Q, B)$ will be closed under the Boolean operations $\oplus$ (XOR) and & (AND). Let's call $\mathfrak{P}(Q, B)$ as set of **Verbs** and the resulting values as **actions**.

**Definition 8.** Let's call the mask set $\mu$, as an array of quantities $Q = (q_0, \ldots q_n)$ and an array of actions $A$, where each element of $A$ is an offset from the beginning of mask, as **Noun**



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

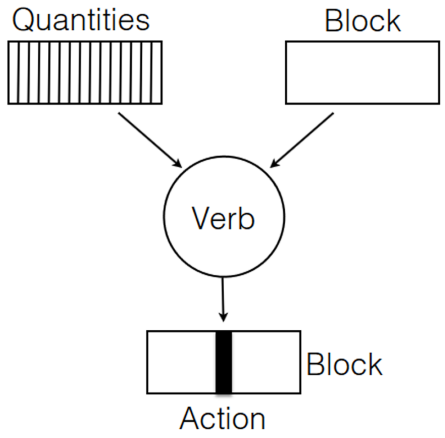


Figure 3: Action of Verb

**Definition 9.** Let's call the Noun set $\mathtt{N}$, Adjectives $(\mathtt{A}_0, \ldots, \mathtt{A}_k)$, and Verbs as **simple class**. Moreover, verb for simple class will be the ingress. Verb will use qualities and blocks as arguments.

We note that an action may be employed, but not on every point of a layer.

**Definition 10.** Let's call a class without any verbs, or adjective, as an empty class. An empty class will contain only a noun.

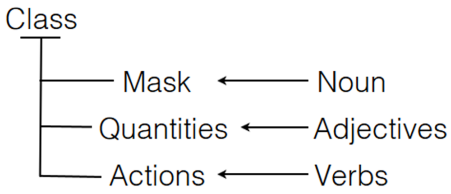


Figure 4: Simple class structure

Under a simple class six operations can be performed:

**Noun Specialization** Create a new class with a new noun $\mathtt{N}'$ which equals to $\mathtt{N}$ with additional adjective condition $\mathtt{A}_q = 1$.

**Noun Argumentation** Create a new class with a new noun $\mathtt{N}'$ which equals to $\mathtt{N}$ with additional verb condition $\mathtt{V}_q = 1$.

**Verb Addition** Add to class a new verb $\mathtt{V}'$ which equals $\mathtt{V}_l \oplus \mathtt{V}_m$.

**Verb Multiplication** Add to class a new verb $\mathtt{V}'$ which equal $\mathtt{V}_l \,\&\, \mathtt{V}_m$.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

**Adjective Addition** Add to class a new adjective $\mathtt{A}'$ which equals to $\mathtt{A}_l \oplus \mathtt{A}_m$.

**Adjective Multiplication** Add to class a new adjective $\mathtt{A}'$ which equals to $\mathtt{A}_l \,\&\, \mathtt{A}_m$.

Due to finiteness of the quantity of verbs and adjectives in the class, it is possible to introduce the specific bijective function $\Phi$, such that it will display a class of operation to the layer.

If for a new adjective creation there is a unique operation, then the creation of the new verb has an internal ambiguity. It is because for a new verb we need to find a new point of action. In order to solve the ambiguity, we need to show one or more active adjectives, which are changed by the verb action point. The action point should be an argument of all active adjective and should not be an argument of all others.

**Definition 11.** Let's set classes $\mathtt{C}$ and six operations between them. There is a fixed finite array of elements $(C_0, \ldots, C_N)$, which are called as basis. Now we can introduce the subset of classes $\tilde{\mathtt{C}} \subset \mathtt{C}$, which can be created from the basis, by a finite count of operations. And we can introduce bijection $\Phi$ from $\tilde{\mathtt{C}}$ to the natural number set $\mathbb{N}$. Let's call it as **overclass mapping**.

The specific form of $\Phi$ mapping is not important for the further consideration, note that it is invariant during the whole life.

## 3 Layers Working

The mind system is made of stack and common objective memory. First, let's describe layers and their interactions. Null or signal layer is responsible for working with sensory and motor neurons. They excite and inhibit under the influence of the external environment. The current job does not put a purpose to solve the problem of response, so we will not consider the generation of egress signals.

Let's consider layers by induction. There are three layers, $i+1$, $i+2$ and $i+3$. For brevity let's call them as 1st, 2nd and 3rd, and remembering that they are not absolute numbers of layers.

On the 1st layer, there seems to be an excitement of elements. Thereafter, regularities are detected on a layer by overlaying noun's masks. Let all possible nouns be detected and the whole layer will be covered by blocks. It is not necessary to block all points of layer, some points can be free from blocking.

After this, on the 2nd layer, masks are forcibly excited of the same nouns in the same order, but without duplicated elements and gaps. *i.e.* if on the 1st layer nouns created in a sequence of $(A, B, B, B, C, C, D, A, C)$, then at the 2nd layer there will be a sequence of $(A, B, C, D, A, C)$.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

On the next stage, the excitement state will be transmitted to the 3rd layer, also by detecting the noun's masks. Let the sequence $(A, B, \ldots, A, C)$ be responsible for the noun $E$, and this sequence be called **sentence** of $E$. It is the simplified version, in any case the nouns which will be important of 2nd layer, are those which were covered in the ask of the 3rd.

**Requirement 12.** Masks, which are forcibly excited on the layer, will not be detected on it.

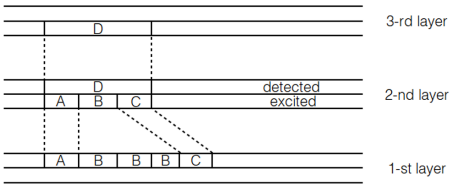


Figure 5: Object detection

**Definition 13.** Let's call the simple class example which has specific qualities and actions as simple **object**. Every simple object is linked to the simple class, on which it acts. In the current examination objects were created on 1-st, 2-nd and 3-rd layers.

**Definition 14.** Let's call the set of the simple classes and simple objects, which are denoted by the classes, as 0-level **context**. Let's call the pair of simple class and 0-level contexts as **complex class** 1-level.

Context is made on the 3rd level, made in the 1st and 2nd levels, in which further changes are contained. Now the 2nd layer controls the classes of context and 1st layer controls the objects.

Let's consider the behavior of objects of the 1st layer. They are in one context of 3rd layer, so they are connected together. In the beginning, they load their qualities from appropriate blocks of the 1st layer. In pseudocode:

**for all** object $o$ in context **do**
  **for all** adjective $a$ in $o$ **do**
    let $q_a$ is quality of $a$
    let $B(o)$ is block of $o$
    $q_a \leftarrow a(B(o))$
  **end for**
**end for**

Then placement of verbs happens. Let the verb $f$ acts on the class $D$, and its quantities $(q_0, \ldots, q_n)$ be its arguments. Let $B(d)$ be the block of the object $b$. Then we should find classes with required quantities in context. Let class



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

$C_b$ contain such quantities. Then objects of the class $C_b$ act on each object $d$ of class $D$:

$$d \to d' = f(B(d), b_i) \text{ for each } i \in (0, \ldots, k)$$

In pseudocode:

**for all** object $d$ in context **do**
  **for all** verb $v$ in $d$ **do**
    let $a_v$ is action of $v$
    let $B(d)$ is block of $d$
    let qualities $(q_0, \ldots, q_k)$ are linked with $v$
    **for all** object $b$ in context **do**
      let $C_b$ is a class of $b$
      **if** $C_b$ has qualities $(q_0, \ldots, q_k)$ **then**
        $a_v \leftarrow v(B(d), b)$
      **end if**
    **end for**
  **end for**
**end for**

It is very important to note that if verbs change the state of the block, instead of adjectives which change the objects, then it affects the following verb's actions.

## 4 Context Working

Let's introduce a series of definitions by induction:

**Definition 15.** Let's call a pair from the set $(N-1)$-level a complex class and the set $(N-1)$-level a complex objects as $N$-level **context**. Let's call the pair from a simple class and set of pairs of $(N-1)$-level complex objects and $(N-1)$-level contexts as $N$-level **complex class**.

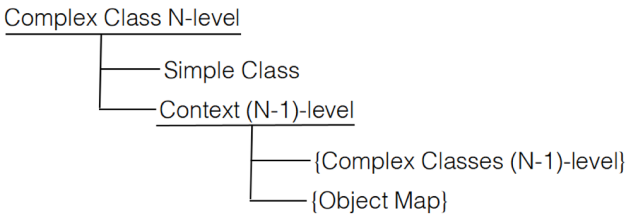


Figure 6: Complex Class

In previous section there was a mind stack which contained 3 layers. Later we will consider a mind stack which will contain 6 layers. It this case



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

verbs actions can be applied on 4 lower layers. Let us understand what will happen to them sequentially.

Suppose that required masks were identified at the lowest layer, and after that masks were found on all layers, including the 6th layer. Now let's consider what happens at the 2nd layer. Because contexts are found on the 4th layer, the 2nd one has objects with adjectives and verbs. So excitation will spread onto the 2nd layer. The points on the 2nd layer are responsible for classes on the 1st layer. And due to existence function $\Phi$, which connects data with class operations, modified classes will act on the 1st layer rather than original ones.

Since basic classes were in a global context, they were chosen from common memory of the system. So the modified classes on 2nd layer are already local ones for the their contexts. The local classes are placed into context and completely replace the lower layers for global classes with the same masks. Further operations on local classes only fulfill this context.

The next stage begins when the 5th layer happens. After this, it is possible to modify objects on the 3rd layer. So what happens when object with attached context are changed? There are 2 options: current context is modified or new empty context is created. Mentioned above, every object has qualities and actions. So it is possible to have two situation, each with its own named:

**Static** New context will be created if any quality on object is modified. Object with old attached context be placed into memory and will be available when specific qualities combination is met. item[Dynamic] New context will be created if any action on object is modified. Object with old attached context be placed into memory and will be available when specific actions combination is met.

In every mind system only one version of the system happens. It does not change during the lifetime [MM95]. Such a case is called **option**, this option is between static and dynamic. Let's call it **context option**.

In the layer system, described in the previous section, the upper layer creates classes on the lower one. In context memory the reverse process takes place, where upper class is modified by a set of classes from the lower level. For that there are so-called **spontaneous** class operations. Every mind system has 3 spontaneous operations, one for nouns, one for verbs, and one for adjectives. Let's introduce appropriate options:

For nouns:

**Ratio** Noun Specialization operation is spontaneous.

**Irratio** Noun Argumentation operation is spontaneous.

For verbs:



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

**Logic** Operation XOR for verbs is spontaneous.

**Ethics** Operation AND for verbs is spontaneous.

For adjectives:

**Intuition** Operation XOR for adjectives is spontaneous.

**Sensorics** Operation AND for adjective is spontaneous.

Now it is possible to show how spontaneous operations work. First, we introduce some notations. Let's represent an object by the symbol as $O$ and classes by $C$. Upper index of classes and objects will represent a level. Apostrophe in classes $C'$ will represent what class is modified with the respect to the global class of $C$.

The 3rd layer has an object $O_1^3$ with a class $C_1^{3\prime}$. The object $O_1^3$ has context within which there are objects $\{O_i^2\}$ and the classes $\{C_0^{2\prime}, C_1^{2\prime}, \ldots, C_N^{2\prime}\}$. The task is to modify the class $C_1^{3\prime}$ and the $\{C_0^{2\prime}, C_1^{2\prime}, \ldots, C_N^{2\prime}\}$ so that the methods (verbs and adjectives) of $C_1^{3\prime\prime}$ could create lower level classes, and to modify lower level classes, so they could create objects $\{O_i^2\}$. So the objects have to be immutable and classes in context could be modified. Later we will describe the algorithm step by step. But it should be noted that the algorithm creates not the only result, but many available results, and only the final step will choose the most appropriate one.

**Algorithm Begin**

**Step 1** Described previously, there is a bijection $\Phi$ between natural numbers and class, based on a global basis. Now consider the inverse transformation of $\Phi^{-1}$, which presents the class as ones and zeros. Let call it as binary representation of class. We will give the table a conventional example of the changes taking place.

| | |
|---|---|
| $C_0^{2\prime}$ | $10101010\ldots10$ |
| $C_1^{2\prime}$ | $11101010\ldots11$ |
| $\ldots$ | $\ldots$ |
| $C_N^{2\prime}$ | $10101110\ldots00$ |

**Step 2** Binary representation of all classes in context is modified by using noun spontaneous operation with condition that modified classes should describe all of its objects $\{O_0^2, \ldots, O_M^2\}$ in same context. The aim is to find same parts in all classes of the binary representations. All later steps will only be about the differences of binary classes representations.

| | | | |
|---|---|---|---|
| $C_0^{2\prime\prime}$ | 0101 | $001\ldots10$ | 10 |
| $C_1^{2\prime\prime}$ | 0101 | $101\ldots11$ | 10 |
| $\ldots$ | $\ldots$ | | |
| $C_N^{2\prime\prime}$ | 0101 | $010\ldots00$ | 10 |



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

**Step 3** Let's move to the class $C_1^{3\prime}$. It has some adjectives, each of which take on a binary representation, and as a result gives meaning of quality. It is necessary to create new adjectives via adjective spontaneous operation, which will distinguish all different binary parts through $C_1^{3\prime}$ qualities.

| | | $q_0, q_1, \dots$ | |
|---|---|---|---|
| $C_0^{2\prime\prime}$ | 0101 | $001 \dots 10$ | 10 |
| $C_1^{2\prime\prime}$ | 0101 | $101 \dots 11$ | 10 |
| $\dots$ | $\dots$ | | |
| $C_N^{2\ \prime\prime}$ | 0101 | $010 \dots 00$ | 10 |

**Step 3.1** An additional step will be taken into action, if all possible algorithm results are considered invalid. In this step it is possible to introduce new adjectives which directly represent bits from layers to new qualities.

**Step 4** Before this step, let's remember that all classes in contexts were created from global contexts, by a series of verbs transformations. It is very important to note that verbs from $C_1^{3\prime}$ were used sequentially, when action from one was argument for another. Now it is necessary to do so what lower class will be created faster.

Let there be two verbs $V_1 \to A_1$ and $V_2 \to A_2$, which create actions $A_1$ and $A_2$. Also $A_1$ is an argument for $V_2$. Therefore the verb $V_1 \circ V_2 \to A_2$ leads to the required value of $A_2$. A subtask of this step is to find a way, how it is possible, by using spontaneous verb operation, to create new verbs without intersections, by its arguments, and the ability to create classes for its contexts or parts of them.

**Step 4.1** is also an additional step for invalid results. Let there be context attached to $O_1^4$ and it be contained in the context of $O_1^3$. And let this context detect sentences where the quality from $O_2^3$ has always the same value with the bit $x$ from the block of $O_1^3$. Then in the mind, after a few tries, the system will create a verb, which connects quality to action over bit $x$.

**Step 5** All previous steps gave some set of possible realizations of class $C_1^{3^?}$. But the final version will be chosen after a uniqueness and an accuracy check. The memory system waits when the object $O_1^3$ will be loaded to layer. The upper layer will contain the sentence with $O_1^3$. The lower layer will contain a set of sentences with the set of that class. If all lower classes are in proposed realization, then it will be chosen as $C_1^{3\prime\prime}$ instead of $C_1^{3\prime}$.

**Algorithm End**

The process is simplification of memory system, it reduces the whole classes count and creates new verbs and adjectives. It called as **abstraction**. It is important to note that division into spontaneous and non-spontaneous operations required for solving problems with infinite looping. If count of nouns, verbs and adjective is finite then system will have finite count of there derivatives by spontaneous operations. The different combination of context



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

creation and spontaneous operations options give 16 variants of behavior of the entire system.

## 5 Structure of Input-Output System

Let's analyze/look at the input-output structure in detail. As we know, there is a vast amount of signals from muscles and receptors; the overwhelming flow can hardly be directly processed by human consciousness. For primary processing of the flow there are *classical conditionings*. They can be expressed in the following chain:


Figure 7: Classical conditioning

The system of all classical conditionings is *the dynamic stereotype*. The dynamic stereotype plays multiple roles:

- It protects the consciousness from the huge flow of information from all the neurons in the body.
- It takes care of actions where comprehension is not necessary, including simple reactions like stepping, saying simple sounds, or "hit or run" reactions.
- As a result of its work, the consciousness, instead of receiving certain signals, receives whole images, which are recognized at the level of the dynamic stereotype.

Let's try to construct a classical conditioning from the suggested model. For this purpose a special class where all verbs are connected to its qualities will be suitable. We will call it **the conditioning class**. A conditioning class is *a deterministic finite automaton*. A conditioning class should not only react to stimuli, but it should send signals to the consciousness. Therefore a conditioning class is a self-acting class, which processes signals from receptors and reports about some combinations of the signals to the consciousness.

Therefore, all input-output can be divided into two segments: input segment and output segment. The input segment receives all signals from already working conditioning classes; the output segment is a space for generation of conditioning classes, which will be loaded to dynamic stereotype. Also it will be possible to determine on the output segment whether a class is suitable for the dynamic stereotype or not. If a newly generated class has a verb which requires any external qualities, then the class will not be



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

conditioning and will not upload to dynamic stereotype. The algorithm of a working conditioning class is the following:

**Algorithm Begin**

**Step 1** A conditioning class $R$ uploads to dynamic stereotype.

**Step 2** $R$ receives access to motor and sensor neurons, also $R$ receives a private area in signal layer for its output.

**Step 3** $R$ detects images from sensor neurons, influences motor ones, outputs data to signal layer of the stack.

**Step 4.1** If $R$ detects all images successfully, after some time it will be unloaded from dynamic stereotype by a next command from the stack.

**Step 4.2** If $R$ does not detect a signal from sensor neurons (*i.e.* its mask failed to apply), it will be forcibly unloaded, and will be interpreted by input of signal layer as **a failed** one. Processing of a failed class will be described in the next section in detail.

**Algorithm End**

The main function of the dynamic stereotype is automated processing outside the consciousness. However, there is one more capability of it. After all, a human cannot only pronounce sounds, but he can think about how he pronounces sounds. This second option is called *internal speech*.

Let us examine internal speech in more detail. A human can play out actions in his head, cause sensations, conduct mental operations, along with an internal monologue. These mental actions are absolutely similar to ordinary ones; they can cause the same recoil, but do not involve any muscles. Internal speech is that of a conditioning class which receives all signals from receptors, but all its out-going actions are connected to the input of the signal layer, not to muscles.

After examining this ability, we have come to the second feature of the output – **speech direction**. It turns out that the output segment of the signal layer consists of a set of pairs – conditioning class and speech direction. Speech direction serves for determining where a conditioning class will be uploaded for work. Speech direction has two options: **the internal speech** and **the external speech**. In the case of external speech a conditioning class will directly work with signals from muscles and receptors.

Now let us remember that a mentally healthy human being has two qualities:

- First of all, a human has only a single internal voice;
- Second of all,, the human is able to distinguish an internal voice and his fantasies from the reality.

Therefore, the internal speech has two important qualities:

**Requirement 16.** The internal speech can process only a single class.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

**Requirement 17.** The Output of internal speech is directed to a specially selected area of the signal layer, that is not intersected with signals from external speech.

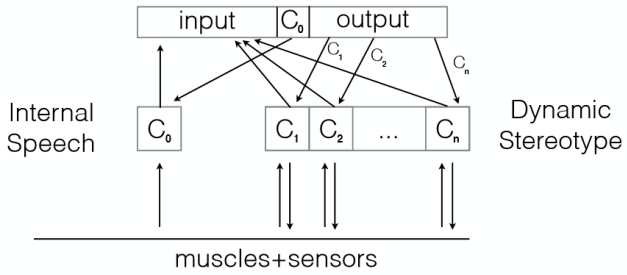


Figure 8: Input and output

Internal speech performs a number of important functions which extends the model:

- Internal speech permits to mentally conduct operations that are not spontaneous for the memory. For example, it is possible to create a conditioning class that conducts an operation of verb addition XOR for an ethical organization of the memory. Using this class does not substitute for the operation, but a human will have the possibility to remember some rules that need to be done in certain circumstances,which will give the right result.

- Internal speech also serves for control of the memory and the stack. By continuously uploading some information to the signal layer, the human is able to process the information as detailed as possible.

## 6 Decisions

The memory model has the algorithm of context packing that is able to create more complex classes from a set of lower ones. In most cases the algorithm is multivalued, *i.e.* it can give multiple results. It has two types of multivaluency:

- Firstly,, multivaluency of operations. This is when it is possible to create multiple combinations of nouns, verbs and adjectives that are able to create all lower classes in some circumstances.

- Secondly,, multivaluency of data. It is when its impossible to determine a point of action by the suggested adjectives.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

In order to describe the multivaluency, let us suppose that using the packing algorithm creates all possible classes. We denote the whole memory tree in an initial state as $T$. In addition, the $\{p_i T, i = 1, 2, \ldots, N\}$ set will contain all possible packed trees with all possible packed classes. We will have marked with $p$ any changed tree $T$, and index $i$ enumerates all possible variants. In particular, the packing algorithm acts as the function

$$T \to \{p_i T\}$$

We will call any change **a patch**. This term presumes that packing generates $N$ different patches.

Every patch is described by packed class $\tilde{C}$ and its context. The $\tilde{C}$ class has more nouns and adjectives than the original $C$, but class's context decreases. Moreover, the creation of a set of $\{\tilde{C}_i\}$ classes with difference in an action point, is possible. We mark it as a possibility, but the specific implementation of multivaluency will not be important later. It's Important is that there is a finite set of all values in multivaluency.

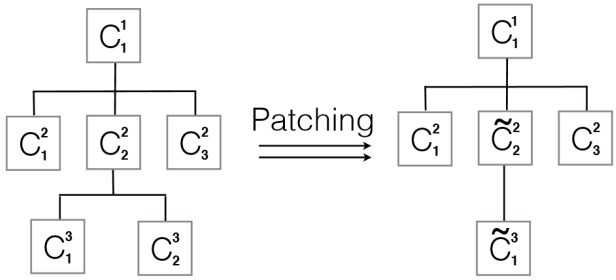


Figure 9: Patching

However, packing can be applied not only to the original $T$ tree, but to any earlier packed tree. Let $p_i T$ be packed again, and $\tilde{C}$, changed at first stage, will be changed one more time. The result of the packing will be a set of memory trees $\{p_j^1 p_i T, i = 1, 2, \ldots, M\}$. It will be the second level patch. We will denote $n$-level patch as a $\pi^n T$.

Consequently, because all patches has all different classes that are don't intersect with each other, all possible patches $\pi T$ will form a structure like a tree. We will call the tree **the decision tree**. The original state of the memory without any modification is a root of the decision tree; leaves of the decision tree are final states of the memory with all possible modifications, let us denote them as $\hat{\pi} T$.

Let us examine the making of a decision. Let's say, $\sigma$ was the initial state of the signal layer's input. We may say that the $\sigma$ applying to the memory tree $T$ led to the decision tree generation, because $\sigma$ exactly had additional data, that helped to simplify the original state of the memory $T$. It proceeded by the following algorithm:



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

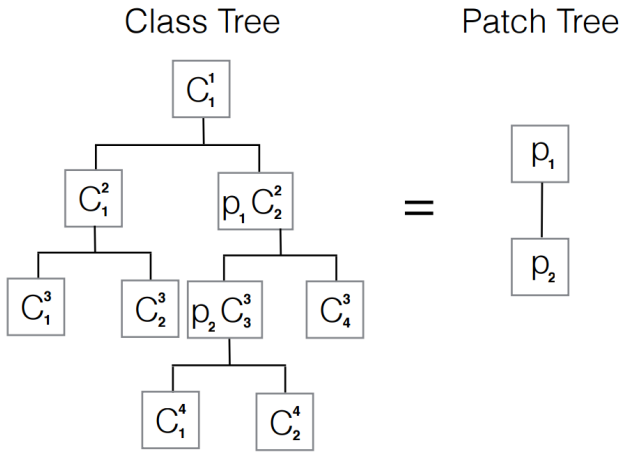


Figure 10: Tree of patches

**Algorithm Begin**

**Before Steps** The stack is clear, but the memory has a state as $T$.

**Step 1** Data is coming to the input of the signal layer. The signal layer has a $\sigma$ state; all levels of the stack are filling with decoded data.

**Step 2** During the process of filling upper layers', the memorized classes are addressed, that are modified according by the information from the $\sigma$. The memory tree is now set to $\tilde{T}$.

**Step 3** The Modified state of the memory tree $\tilde{T}$ is packed by spontaneous operations. As a result of packing, there is a set of possible memory trees $\{\hat{\pi}_i T, i = 1, \ldots, N\}$.

**Step 4** Each of the finished $\hat{\pi}_i T$ memory trees according to the signal, generates the segment's condition, which follows the regular algorithm of creating a response. You can say that the $\sigma$ signal creates more than one response, but instead a host of possible $\{\kappa_i, i = 1, \ldots, N\}$ responses.

**Step 5** The algorithm is branched here.

**Step 5.1** If $N = 1$, *i.e.*, meaning the memory packed in the only possible way, then the memory will be fixed. A fixing of the memory is replacing the original $T$ state by the changed one $\pi T$; and the $\kappa$ response is generated on the output of the signal layer. The algorithm is finished here.

**Step 5.2** If $N > 1$, there is a set of possible options. Because it is impossible to generate multiple responses at one time, the following happens. All possible $\{\kappa_i\}$ responses from all possible packed trees will come to the signal layer again, but not to the usual output. They all together will come to the special zone of input; and the memory will be $T$ again. Let us call it the special zone, which is responsible for resolution of multivaluency, as an **Ego**. The algorithm is recursively iterated here with the new data; the



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

return from the iteration will be to the next steps, denoted as post-steps.

**Step 5.2 post 1** If the result of the step 5.2 is the $\kappa_l$ response, then the final state of the memory will be $\pi_l T$.

**Step 5.2 post 2** In case none of suggested responses were chosen at step 5.2, but during the data processing the new $\pi^{(1)}T$ patch was created and the patch gave the only one response; then the patch will be chosen as a final state of the memory.

**Step 5.2 post 3** If there is no single response, all data is dropped and a response is not generated.

**Algorithm End**

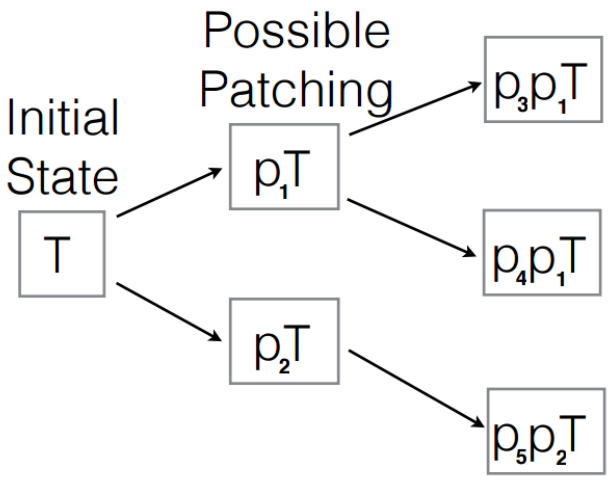


Figure 11: All possible versions of memory tree after patching

## 7 Model Training

Taking a more practical approach to the task will allow us to create a model. Let us look into a class structure that should contain all data and operations, where operations should be able to convert to data and vice versa. . It is shown in the following, more precise, picture:

One of the basic model requirements is the possibility to determine the same actions and qualities in different classes. The simplest way to solve this problem is to attach a global noun with its own mask to each action or quality. Thus, we will be able to reach uniform interaction by connecting same masks of different classes' objects. It could be described by the scheme:

Therefore, we should formalize these two processes:

First, the *Op* operation on the following predicates:

$$NewPredicate = OldPredicate(AND/XOR)OtherPredicate$$



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

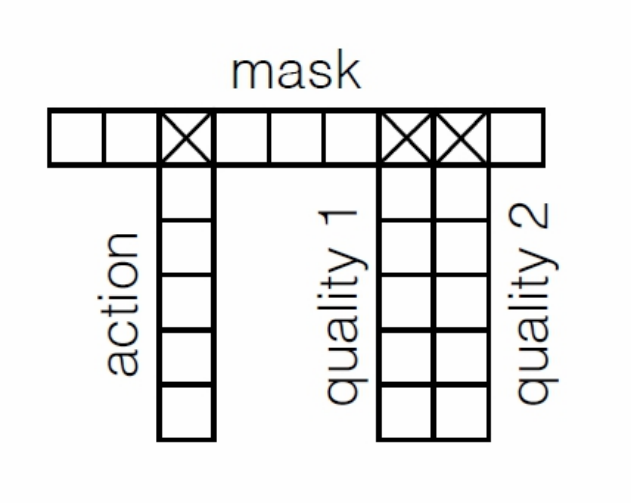


Figure 12: Class in 2D

where $OtherPredicate$ is described by the $mu$ mask of its noun to change predicatesof current classes.

Second, creation of a new mask for new qualities and actions necessary for extension of current classes.

Let us remember that a class can be in three states, when it is child on the stack, when it is parent on the stack and when it is stored in the memory. Parent state is the simplest one, at this moment class's object reads qualities from the lower layer and applies actions to them. It was described above in detail.

The child state is more difficult than the parent one, it implies that class itself can change. Operations can be applied to the qualities of verbs and adjectives; new actions and qualities can be created. In addition, the child state should be responsive to the static-dynamic option to get access to different variants of nested contexts in the memory. The easiest way to express it would the following:

1. Mask of the noun
2. Adjectives in binary format
3. Verbs in binary format

What is important here, is the ingress action points. Action points are parameters that can be used to change classes by writing values into them. Every action point has two parts, operational bit and mask of the argument. If operational bit = 1, then $AND$. If operational bit = 0, then $XOR$.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

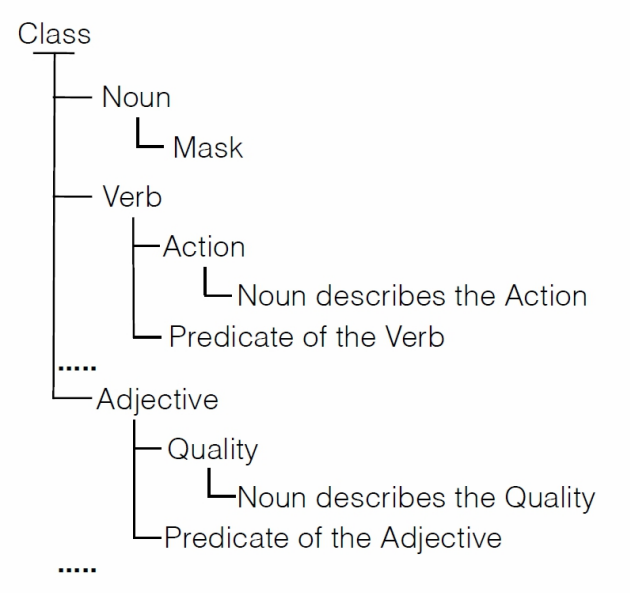


Figure 13: Scheme of Class

The third state is class's predicates serialization. Adjective's serialization can be carried out as follows:

1. Arguments' offsets
2. Operations in a binary format
3. Mask of the noun that describes the quality

And verb's serialization as follows:

1. Masks of nouns that describe adjective
2. Operations in a binary format
3. Offset of the result action point

For the ordering of binary operations we will use Zhegalkin's polynomial representation [Zhe27]. Thus, all operations on predicates are stored as data.

However, the actual difficulty is not classes' predicates storing but packing them in the memory. Its meaning is that a parent class should be converted to generator of one's nested classes. We have to order children's masks to allow operations from the parent class create all required operations. Moreover, there should be multiple variants of nested class's recombinations, if



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

possible. It is possible because the nested classes have already been created from the global ones. However, it was a multiple-step process.

The most suitable option for development of such methods would be the genetic algorithm [Fre02]. The following parts are required to define the Genetic Algorithm:

1. spontaneous operations described earlier; they mix predicates of parent class for the best match to the test data;
2. fitness function that takes test data and runs it through modified parentclass to determine the one that generates more nested than others;
3. test data that is various recombinations of masks of the nested classes.

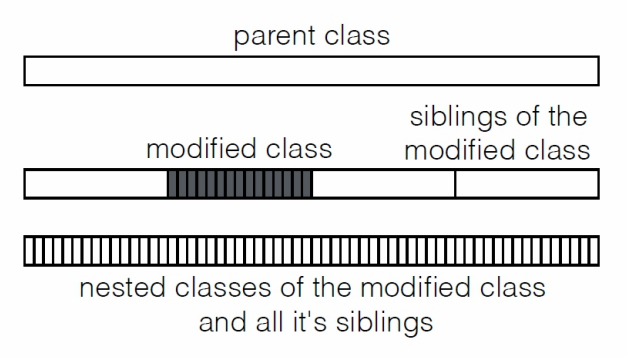


Figure 14: Place of the modified class

The first two parts are already determined; now we need to sort the required test data. The main difficulty here is that fully functional performance requires test data of all siblings of parent class and their nested and subnested classes. Therefore, generation of test data should be intellectual; simple recombination of the nested class would not work.

It is suggested that system of neural network be used/ We will call it an Imaginator. Imaginator is a Generative Adversarial Networks [GPAM$^+$] and has two parts: a Generator and Discriminator.

The working loop of Generator $G$ is in the following algorithm:

let $C$ is class
let $T$ is tree data for evolution testing
**for all** class $D$ in context of Grand-Parent of $C$ **do**
  let $R_0$ are all children of $D$
  let $G(R_0)$ is recombination of $R_0$
  $G(R_0)$ MUST be covered by mask of $D$
  $G(R_0)$ MUST pass Discriminator
  put $G(R_0)$ to $T$



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

**for all** unique class $P$ in $G(R_0)$ **do**
let $R_1$ are all children of $P$
let $G(R_1)$ is recombination of $R_1$
$G(R_1)$ MUST be covered by mask of $P$
$G(R_1)$ MUST pass Discriminator
put $G(R_0)$ to $T$
**end for**
**end for**
$M(C)$ are all classes modified by Evolution
apply $T$ to $M(C)$ to get new modified contexts
choose the most appropriate one

it is required that Generator $G(R)$ output corresponds with the mask of each parent class. However, it is not the only requirement. It is necessary that the Generator "could" pass the Discriminator. The Discriminator must ensure that Generator's output is realistic. For this purpose, the Discriminator learns on real data, specifically on combinations of nested classes that appear on the stack during work. The full learning process can be shown as follows:

1. Discriminator gets data for self-learning from the stack and trains Generator.
2. Generator makes test data for Genetic Algorithm.
3. Genetic Algorithm generates patches.
4. If patches are applied to the memory tree, the data from the patch are considered as good.
5. If patches roll back, the data from the patch are considered as bad.

As the result, we see a chain of the Imaginator and the Genetic Algorithm that creates patches for the memory tree. In general, the process is as follows: the Discriminator takes recombinations for learning from the stack and trains the Generator; The Generator creates test data for the Genetic Algorithm; the Genetic Algorithm creates changed parent classes that are packed to patches.

In addition, the Imaginator learns itself during the patch creation process. Apart from the standard learning process of the Generator and Discriminator, described above, there is an additional one. Let it be the set of patches $\{P_i\}$. If patches are applied to the memory tree, it would be a changed memory tree. This tree is then tested on the real data derived from the stack. If during the work flow the system reserves a positive motivation, then the changes are fixed. If the system does not receive a positive motivation, the changes are dropped. However, besides the positive motivation



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

that reinforces the memory, there is pain. If the applied changes result in the pain syndrome, then $\{P_i\}$ patches are not only fixed, but also go to the Discriminator's input as negative; and the Disciminator learns to avoid them. Thus, the system is configured to prioritize pain avoidance over a motivational reinforcement.

Prospect theory [KT79] assumes that people intuitively analyze possible events in the wrong way. Low probabilities tend to be overvalued and the high ones tend to be undervalued. This effect should be observed in the model because the Generator creates recombinations of nested classes excluding the probability they occurrence in the real data. In addition, the overvaluation of negative possibilities should be observed because the pain effects are applied directly to the Discriminator.

For the sake of completeness, we should introduce one more neural network, the Detector. It is also GAN that contains two parts. The Detector's discriminator searches repeating sequences in the stack to create new masks. The Detector's generator creates new unique sequences, which would never appear on the stack, to mark new qualities and actions. The Detector's generator learns with the help of its Discriminator.

To give the final definition of the system with Reinforced Learning, we have to describe the input data and its ways. The continuous working process should have two types of data: ingress data for routine operations and control data for approval/rejection of generated patches.

Ingress data come from two sources:

1. Signals from motor and sensor neurons with information from the external world. This is quite a simple type of information and the system considers it as external. It is called *feeling*.

2. Signals about total changes in the sets of applied and dropped patches. It has no precise knowledge about the information from the patches, but only indicators that show whether a patch was applied or dropped and in what order. This type of information is considered as internal. It is called *emotions*.

The control data do not pass into the model directly. However, they affect the learning process. There are two types of control data:

1. Pleasure. If the system receives pleasure, it applies a patch and reinforces the Generator.

2. Pain. If the system receives pain, it drops a patch and reinforces the Discriminator.

It should be noted that if ingress routine data come into the system continuously, the control data come sporadically. Most of the time the patches are created without any control data, the control data affect the final result only: whether a patch is going to be applied to the memory tree or not.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

## 8 Dynamics

All the previous sections described only one step in the whole system. However, the system should work continuously. Let us introduce the stack frequency. Obviously, the memory works with the same frequency. The memory packer and the Imaginator will work with proportional frequency. However, it is true not for all the system components. The Detector's frequency and operation sequence are not directly connected to the stack frequency.

To conduct further analysis, let us remember that the brain consists not only of neurons, but it is also affected by hormones. Let us introduce two hormones: a happiness hormone and a sadness hormone. The hormones are excreted proportionally to applied or dropped patches. Let an applied patch $P_1$ excrete $H(P_1)$, and a dropped patch $P_2$ excrete$S(P_2)$. It is required to ensure a self-balancing model. If the process runs smoothly and all patches are applied one by one, the model needs a motivation for new data input. If the process fails and all patches are constantly dropped, the model needs a motivation to do something absolutely new that has never been done before.

Let us introduce two frequencies for the Detector. $\omega_{lG}$ is the learning frequency of the Detector's generator and $\omega_{lD}$ is the learning frequency of the Detector's discriminator:

$$\omega_{lD} \underset{\sim}{\propto} \sum_{timeperiod} H(P_1)$$

$$\omega_{lG} \underset{\sim}{\propto} \sum_{timeperiod} S(P_1)$$

where the sum is the total hormone accumulated over a certain period of time. Obviously, if the Detector's discriminator starts finding new stable sequences and creating new nouns, the number of errors should increase.

## 9 Conclusion

This model describes the process of understanding the human speech, but now it has the following problems:

- There is no strict experimental basis, which could show how different people process the information.
- Practical implementation may be difficult because there is no clear understanding of the model's neural network structure.
- The model has no hypotheses or ideas on how to start its implementation from scratch and the newborn reflexes role.



PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1576v3 | CC BY 4.0 Open Access | rec: 20 Nov 2017, publ: 20 Nov 2017

## 10 Acknowledgments

I thank Ilya Kruglov for editing of this manuscript.

## References

[Fre02] Alex A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, 2002.

[GPAM+] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks.

[Jun71] Carl Gustav Jung. *Psychological Types*. Princeton University Press, 1971.

[KT79] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47:263–291, March 1979.

[MM95] Isabel Briggs Myers and Peter B. Myers. *Gifts Differing: Understanding Personality Type*. CPP, 1995.

[Zhe27] Ivan Ivanovich Zhegalkin. On the technique of calculating propositions in symbolic logic. *Matematicheskii Sbornik*, 43:9–28, 1927.