RéSUMÉ. This article should be easy to understand for anybody and is meant to prove that I proposed new kind of operation system based on wiki-like or graph-like structure is

1-more natural, thus easier to learn
2-more efficient on existing tasks in terms of human time spent
3-can handle new kind of tasks

Computer task is data transformation. It is improbable that current paper-like handling of data is the best way. I would like to show that current computers provide a much more natural and useful way of handling all-purpose data. The main idea is that one should store information in a structure as natural as possible, so that user does not have to give efforts to transform the information (express more, search naturally, write once then just reference ...).

I am not sure if I can claim any authorship of following ideas, since one can never be sure whether an idea existed before and what actually helped one to come up with this idea. The only purpose of this paper is thus the pure desire to make progress of thought, by starting the discussion and construction of crowd sourced operation system based entirely on idea of graph databases.

I cannot provide the reader with infinite details and precision, thus I leave some uncertainties to be cleared by the reader himself for pleasure.

My main inspirations for this more natural operation system were: Graph database, Wikipedia, brain, Lisp, mind-mapping, QED manifesto, CSS 3, WikiOS.
1. Searching in a stream

Currently, data are stored in stream files, which one have to read from start to end (not taking into account the lists of contents, index or text-highlighting).

Consider the following example. A writer writes novel. A reader wants to know what’s the color of the hair of a particular character from the novel. If the novel is stored in stream like file, the reader needs to search the whole file manually. This is obviously not effective. The process of obtaining the information would be much more effective if the novel was written in a more structured way, so that the reader could use the structure to get the information. The process would be even more effective if the structure was similar to the one used by the reader’s brain for storage, so that the reader would not have to put much efforts into "translating" his question.

Modern e-book readers solve this adding something like a small Wikipedia to the novel, which, in essence, is just an additional table of contents. This is not efficient since this adds work on the side of the writer. Could the writer write the novel by directly creating wiki-of-the-novel? Could a novel be read like a stream of the wiki-pages designed by the author? How fine should this structure be? My answer are "yes", "yes", "as fine as is practical".

The previous example is meant to be provocative and I cannot yet show precisely that what I claim is true, but I hope that the reader himself will admit it after reading the article. (see paragraph: Examples of possible uses) It is easy to reconsider the example in different and more straightforward cases. For example, change "novel" by “history book”, "math article" or "newspaper".

2. Natural data structure

Let’s get back to the term "natural structure". How does one store memories? Certainly not in form of a word-stream or text-stream. To my knowledge, memory has a wiki-like or world-wide-web structure i.e. it is made of "small pieces of information" connected by "hyperlinks" or "associations" and is strongly clustered in sense of number of connections. One may argue that a wiki is missing is "strength" or "importance" or "length" or "first-that-comes-to-your-mind" "of association". However, this is solved by ordering the links, such that first links are the "most relevant" ones. Notice that once one creates a memory in his mind, it is not necessary to input it any more. If new relevant memory come in, one just adds reference to this memory.

How big are these "small pieces of information" in human memory? I say zero size i.e. they are points. The object 'shoe' in a human memory consists of only its 'associations' to other objects. There is no natural equivalent to the current meaning of the webpage in the brain. The web page "arises" when one tries to explain somebody what a 'shoe' is. Thus, it is a communication artifact rather than a real natural structure. So what exactly happens when you want to send a letter to someone explaining what a 'shoe' is? The brain somehow transforms the association structure to a 'paper-like' or 'stream-like' structure suitable for visual reading. Thus, it adapts the information "shoe associations" to the communication channel. This is not necessary in the case of digital messages, since they can carry 'association structure' (the channel is much broader). Thus, why not to store information in computers the same way as in the brain and attempt to translate to "stream-like" structure only at the end, when displaying on computer screen?

3. Graph database

A graph is defined as a set of vertices and their interconnections. A directed graph is a graph where the edges are oriented. The orientation property will be considered in the following sense : a vertex is a set of "addresses" of vertices that are connected to "A" by edges that are oriented from "A". Thus, if there is no connection from vertex "A" to vertex "B", then "A" is not considered to contain the 'address' of edge to "B", while "B" can still contain 'address' to "A". By address abstract address of object or in other words a pointer to object. Address can be realized by Kernel (see par. :Kernel) for example as address in physical memory.
But how can one store information in oriented graph? In the following example, apostrophe denotes the address to object and three dots denote a continuation in the same manner. The code in this written in the Lisp programming language. To read it, one just needs to remember that bracket content is to be divided by space and that the first member of the bracket "acts on" or "explains" the others.

```lisp
(create-vertexes 'ordered-set)
(create-vertexes 'a, 'b, 'c, 'd, ... , 'y, 'z)
(create-vertexes 'letters, 'clothes, 'shoe, 'trousers)
(create-vertexes 'text1, 'text2, 'text3)
(set-edges 'letters ('a, 'b, 'c, 'd, 'e, 'f, 'g, 'h, ... , 'z))
(set-edges 'text0 ('ordered-set, 'c, 'l, 'o, 't, 'h, 'e, 's))
(set-edges 'text1 ('ordered-set, 's, 'h, 'o, 'e))
(set-edges 'text2 ('ordered-set, 't, 'r, 'o, 'u, 's, 'e, 'r, 's))
(set-edges 'shoe ('text1, 'clothes))
(set-edges 'trousers ('text2, 'clothes))
(set-edges 'clothes ('text0, 'shoe, 'trousers))
```

The task of the "Presenter" program will be to transform the web structure to the suitable format according to the output interface available. In this example a text-only-web-page. Presenter must know the addresses of all the letters, how to display them and need to be instructed what to do with the vertices connected to tag "ordered-set". Let us consider following simple "Presenter" function.

```lisp
(defun Presenter ('address)
  (print "Title=")
  (display-any-text 'address)
  (print "Content=")
  (act-on-edges 'address display-any-text)
)
```

Where the display-any-text function is defined to look for all the edges whose first edge is ordered-set and the rest are members of 'letters and print them.

The result of Presenter('clothes) will be:

- Title= clothes
- Content= trousers clothes

The result of Presenter('shoe) will be:

- Title= shoe
- Content= clothes

To produce more complicated pages, one can of course add tags, like for example equivalent of ordered set tag i.e. one can indicate a father or child relations between different vertices, which presenter should take into account when displaying. There are attempts in the new CSS standard to instruct the web-browser what arrangement of the web-page is preferable regardless proportions of the display.

Notice that one can easily change the letters in previous example to, for example, 2D or 3D pixels. These ones can be arranged into matrices to create 2D or 3D images. It is also useful that one has to input every information only once and later can easily create reference.
4. Math theories and graph database

The set theory is an underlying theory of mathematics. Mathematics is a collection of statements regarding abstract "world of sets". (Definitions as well as axioms in math are definitions of sets. Definitions of sets are just statements of type "This object is a member of the world of sets"). The axioms of the Zermelo-Frenkel theory, apart from defining what the equality of sets is using "in" relation, are just statements of type "If this object is a member of the world of sets then these are too" or "set cannot be a subset of itself".

But this world of sets is just a previously studied oriented-graph database, where edge from "A" to "B" means "A in B". For example, the empty set is a vertex to which no other edges are oriented, but has edges to any other vertex.

It is not surprising that the essence of human thinking (math) has the same character as its underlying hardware (brain). Neither it is surprising that math has been partially successful in describing the laws of physics, since species with badly constructed brain would not survive evolution. It is unreasonable to expect that this math or human-brain approach to the study of the nature is the only possible or best way, in the sense of success to describe and predict behaviour of nature. However, the human mind cannot escape this way of thinking.

Note that a "world of sets" is a database with obviously infinite vertices. But mathematicians do not need it stored as a whole. They just note down some pieces of it or some important truths about it. Thus one can store real mathematical theories in a graph database in a natural manner.

5. One interface

Notice how simple is the interface of Wikipedia. One virtually only needs to be able to search, add, delete or modify articles. It is so simple that a person without any previous knowledge can do that.

Many Linux users do most of the operations on computer using console, because it is usually more efficient. The console is a stream of lines of code. The code is usually just meant to instruct some applications to operate on particular data.

Combining these two approaches together with a heavy use of automatic completions and suggestions, one can create very powerful all-purpose interface. Example: to search, the user would add an edge to the "search now" vertex and the system should add the result edges to the "search results" vertex.

An additional feature to increase the efficiency of searching could be possibility to narrow the search to vertices that are connected to some specified vertex. This, of course, could be further generalized to any kind of set-oriented formula. Example: to search for "dog", one can search the vertices in "animals" without "insects".

6. Programming language of the system

The application, after being called, steps into the list of running applications and the processor from time to time does what the application wants to do. If there are more processors, the application is instructed to enter the emptiest list of running applications on given processors. Can one reproduce this behavior on wiki-like operating system?

To do that, one needs to be able to write programs in a wiki-like structure. If one reexamines the example in the section about Graph databases, a similarity between the structure of the Lisp language and graph database should be apparent. The Lisp type of language is thus a natural choice for such an operating system, since it was created for a similar purpose.

A precise design of the used programming language is not the purpose of this paper, but it appears reasonable to construct this language from the simplest possible operations like: "if-then", "is-connected-to", "not-true", "and", "brackets", "create-edge", "destroy-edge", "create-vertex", "destroy-vertex", "search-text-in-database". The functions should be constructed in a wiki-like-lisp manner,
where the parameters should be passed by adding an edge between the value and vertex representing
the parameter of a function. This should be sufficient to construct more complex commands. The
addresses of executable vertices should be tagged by edge to 'executable' vertex to be recognized
and executed by the presenter.

7. The kernel

A kernel is a software that varies from hardware to hardware. It provides basic functions to
the operating system, such that operating system does not have to be changed from hardware to
hardware.

The kernel should also solve all the hardware-efficiency-issues, including data size issues, automatic code compilation and file-type/encoding issues. This should be purely a task for the computer and should not waste the time of human brains. The human should only solve tasks that cannot be done by any computer program automatically and that is to point out what is important for his purpose. For example, a mathematician points out important truths about the "world of sets".

One would also need special executable vertices called "providers". These are executed by the kernel, when something happens. Most important, provider is "time-change", which should be used to provide multi-tasking and scheduled-tasks possibilities as well as time and date. Second most important is "key-pressed" or similar, that depends what is the main input.

8. New kinds of tasks

Data-mining is easier when the data is stored in a more structured and unified way. It is expectable that much more tasks could by solved by computer for which input data are stored in a such way.

9. Examples of possible uses

(1) Social life
   (a) Social or Cloud systems
       current Facebook-like social networks provide only very limited features
       possibility of creating distributed social networks
       personal data not owned by companies but by users
   (b) Message exchange
       connect all messages from all possible communication channels to one vertex 'inbox'
       reply by adding edge from message vertex to vertex tagged 'reply'
   (c) Instant analyzer of classical stream-like text
       When writing sentence, a computer analyzes the meaning of the sentence using the
       meaning of the previous one. If the writer admits interpretation it is stored. Thus, text
       can be easily analyzed or translated.
       Note that however that it is expectable that with a standardized large enough database
       for communication, one would never need to write classical text to express himself. Thus
       this would effectively replace classical languages, but increase precision of information
       transport and transformation. The transition would still be however smooth since for
       every language one could create instant analyzer of text.

(2) Education
   individual-oriented automated school
   tutoring by walking through the structure
   automatically generated lessons generated from personal graph database of individual
   student
(3) Research
   (a) known-Math-database to spare mathematicians of technicalities
       click to show definition or details of this part of proof
       autoprover
       [ref : mizar.org]
   (b) all-purpose-reasearch wikies
       referencing and citations are not necessary any more
       just adding into existing structure is more efficient