KAPITEL 7 / *CHAPTER 7* [7]
SYSTEMS AND METHODS FOR SOLVING PROBLEMS

DOI: 10.30890/2709-2313.2025-41-07-028

Usually, known approaches are used to describe phenomena, in particular, equations and logical constructions that have previously proven themselves well. That is, on the basis of these proven descriptions and methods, consequences have already been obtained that are consistent with observations and experiments. Such theories are defined as being built on first principles, on which the description of nature is actually based − that is, on repeatedly verified relationships that are repeated under different conditions and with which the scientific community has agreed [1]. In fact, these first principles were formed on the basis of repeated tests and lengthy reflections; they were based on the consensus of knowledgeable people of different generations [2-4.]. The various consequences of new theories obtained in a correct way should not only be consistent, but should also form some kind of coherent picture. All the consequences of applying the equations should not harm each other. When they conflict, destroying the idyll of isolation and consistency − this is an alarm bell. Then everything needs to be revised, and perhaps from the very beginning of the chain of reasoning, calculations and justifications. If there are those who need to check the constructed theory − it is useful for them to suggest a way to check their theory on the part of the creators of this theory, otherwise there will be no trust in the theory. This is the well−known principle of K. Popper: each theory must have important consequences that can be tested experimentally and receive a verdict of nature − correct or not. That is, pass the test for the possibility of falsification. In addition, it is recommended not to introduce new entities into the description (which scientists often do) − Occam's razor.

M. M. Kline [5] believed that "there are many types of reasoning, such as induction, analogy, and deduction." In practice, methods of solving problems are often used, using existing data, that is, induction and analogy, although mathematicians believe that this is a less reliable method that does not provide the desired unambiguity

[7]***Authors:*** *Kuklin Volodymyr Mykhailovych, Sirenka Tetiana Oleksandrivna, Shapovalova Olena Oleksandrivna*
Number of characters: *105225*
Author's sheets: *2,63*

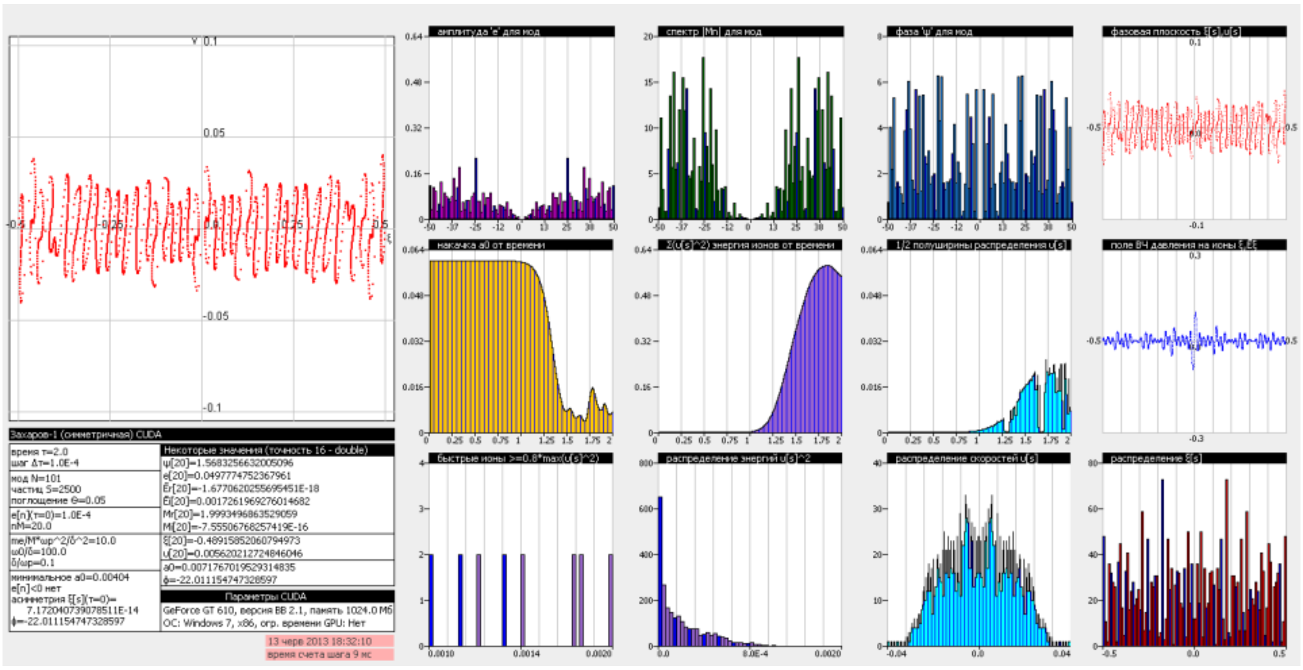
of the conclusion. They believe that it is more reliable to use a system of postulates and axioms to confirm theorems or the rules of deductive logic, identified by Aristotle from the practice of proofs. But in natural science, unfortunately, such postulates and axioms have not been found and they should be invented by looking back at reality. To study problems and phenomena, a certain set of dogmatic statements and rules will be needed, that is, a foundation on which subsequent conclusions are built. For this purpose, practically discovered, guessed, empirically derived relationships between different quantities that determine the nature of processes are used, regularities are formulated, the whole world carefully checks the ability of these discovered regularities to describe reality, finds areas where their application is permissible and where these relationships can play the role of axioms. This set of ideas, regularities, playing the role of axioms in the most general sense, can allow new ideas to be built on this foundation or to check the compliance of various put forward considerations with this set of regularities, which in this case is practically no different from proving theorems. One can focus on experience and multiple repetitions of the same (or similar) consequences from a certain set of fairly uniform phenomena and events. This may also seem to be a certain replacement for the truth of cause–and–effect relationships.

As long as there are no contradictions with observations and experiments, this set is the base, that is, the basis that replaces the axioms. These basic statements and cause–and–effect schemes can be difficult to call axioms, they are practiced precisely as temporary, but axioms. And already on these axioms all subsequent statements are built and ideas are formed, combining them into a paradigm. Often, in addition to proving theorems, that is, reverse deduction, which is the only way to confirm the assumed and answer questions, direct deduction is used. M. M. Kline [5] believed that "there are many types of reasoning, such as induction, analogy, and deduction." In practice, methods of solving problems are often used, using existing data, that is, induction and analogy, although mathematicians believe that this is a less reliable method that does not provide the desired unambiguity of the conclusion. They believe that it is more reliable to use a system of postulates and axioms to confirm theorems or the rules of deductive logic, identified by Aristotle from the practice of proofs. But in natural science, unfortunately, such postulates and axioms have not been found and they should be invented by looking back at reality. To study problems and phenomena, a certain set of dogmatic statements and rules will be needed, that is, a foundation on which subsequent conclusions are built. For this purpose, practically discovered, guessed, empirically derived relationships between different quantities that determine the nature of processes are used, regularities are formulated, the whole world carefully checks the ability of these discovered regularities to describe reality, finds areas where their application is permissible and where these relationships can play the role of axioms. This set of ideas, regularities, playing the role of axioms in the most general sense, can allow new ideas to be built on this foundation or to check the compliance of various put forward considerations with this set of regularities, which in this case is practically no different from proving theorems. One can focus on experience and multiple repetitions of the same (or similar) consequences from a certain set of fairly uniform phenomena and events. This may also seem to be a certain replacement for the truth of cause–and–effect relationships.

As long as there are no contradictions with observations and experiments, this set is the base, that is, the basis that replaces the axioms. These basic statements and cause–and–effect schemes can be difficult to call axioms, they are practiced precisely as temporary, but axioms. And already on these axioms all subsequent statements are built and ideas are formed, combining them into a paradigm. Often, in addition to proving theorems, that is, reverse deduction, which is the only way to confirm the assumed and answer questions, direct deduction is used. Most of them are of zero value in comparison with the array of tasks that we all have to solve. Unfortunately, a significant portion of the tasks do not promise an unambiguous result and we have to put up with this. In addition, these tasks do not have a sufficiently formalized knowledge base. The process of thinking and understanding reality are associated with a continuous comparison of data received from the senses with the gigantic reserves of information about the surrounding world accumulated during the life of an individual. The experience of situations experienced in reality and imagination, together with practiced reflections, form an immediate situational adequate reaction, all types of which are also stored in the human information system, which is largely not realized by them. Such unformalized knowledge can only be obtained through communication with the bearer of this knowledge, in fact, an expert. Therefore, a large number of problems can be solved with the help of expert or recommender systems, the simplest versions of artificial intelligence, the database and knowledge of which were formed with the help of experts with practical experience in solving such problems. From here it is not far to the use of artificial intelligence systems built on other principles, which will be discussed below.

7.1. Integrability problems

But integrability are sometimes very complex and require serious efforts, the use of techniques and methods that will force you to devote a significant part of your life to this activity in order to master them, which the vast majority of people are not ready to agree to. Only to a certain and small extent is D. Deutsch right when he claims that later scientific descriptions cover an increasing part of previously unrelated ideas and make it easier to understand their structure and help find solutions. But more and more often we encounter new theories and descriptions that are not a direct consequence of previous knowledge, but on the contrary, call this knowledge into question. Therefore, mathematicians and theorists in various fields of knowledge are forced to develop increasingly complex means of solution. Examples can be given from the field of theoretical physics, where in order to solve problems of ion heating in plasma wave fields it was necessary to create extensive interfaces for interactive interaction of researchers with the solving device, in particular, a high–performance computer server. In particular, when using such an approach, in the works [6,7] it was shown that the mechanism for the formation of the normal distribution of particles by energy is scattering on field inhomogeneities (the Fermi effect), which turned out to be more important than the mechanism of energy exchange between the field and particles, known as Landau damping, which traditionally explains this phenomenon. But for this, Dr. A. V. Priymak needed to create a diagnostic stand demonstrating the details of the process (see Fig.1).


амплитуда 'e' для мод
спектр |Mn| для мод
фаза 'ψ' для мод
фазовая плоскость ξ[s],u[s]
накачка a0 от времени
Σ(u[s]^2) энергия ионов от времени
1/2 полуширины распределения u[s]
поле ВЧ давления на ионы ξ,ξξ
быстрые ионы >=0.8*max(u[s]^2)
распределение энергий u[s]^2
распределение скоростей u[s]
распределение ξ[s]

```
Захаров-1 (симметричная) CUDA
время τ=2.0
шаг Δτ=1.0E-4
мод N=101
частиц S=2500
поглощение Θ=0.05
e[n](τ=0)=1.0E-4
nM=20.0
me/M*ωp^2/δ^2=10.0
ω0/δ=100.0
δ/ωp=0.1
минимальное a0=0.00404
e[n]<0 нет
асимметрия ξ[s](τ=0)=
    7.17204073907851 1E-14
ϕ=-22.011154747328597

Некоторые значения (точность 16 - double)
ψ[20]=1.5683256632005096
e[20]=0.0497774752367961
Ẽr[20]=-1.677062025569545 1E-18
Ẽi[20]=0.0017261969276014682
Mr[20]=1.9993496863529059
Mi[20]=-7.555067682574 19E-16
ξ[20]=-0.4891585206079 4973
u[20]=0.0056202127248 46046
a0=0.0071767019529314835
ϕ=-22.011154747328597

Параметры CUDA
GeForce GT 610, версия ВВ 2.1, память 1024.0 Мб
ОС: Windows 7, x86, огр. времени GPU: Нет

13 черв 2013 18:32:10
время счета шага 9 мс
```

Fig. 1. Diagnostic stand for calculating systems of integro–differential equations on a computer (400 modes, 20,000 particles). Source [6,14]

A similar diagnostic stand created by him was intended to study the emergence and development of waves of abnormally large amplitude (> 20 meters) in the oceans (see Fig. 2). The disturbances of anomalous amplitude, arising as a result of nonlinear interference of sufficiently intense sea waves (the average amplitude of waves in a rough ocean is 4–6 m, the period is 10–12 s, the phase velocity is more than 20 m/s, the group velocity is half as much, and the attenuation lengths of waves are measured in thousands of kilometers), are a sequence (group) of usually 3 waves, one of which is the largest; the occurrence frequency (ensemble statistics and time) of such a group of waves is one in 10–20 thousand waves. For such long waves (200–250 m), the maximum achievable amplitude before breaking is about 30 m. They most likely occur during the period of modulation instability development (in the range of hundreds of kilometers from the boundary of the wind–driven wave excitation zone, the instability development time is 10 inverse increments, i.e. about 2.5–3 hours). In the book [8] it is shown that the disturbance of the ocean surface arising from the development of modulation instability is similar to a Peregrine breather/soliton. These killer waves, destroying ships and vessels in their path, can propagate for hundreds of kilometers [9]

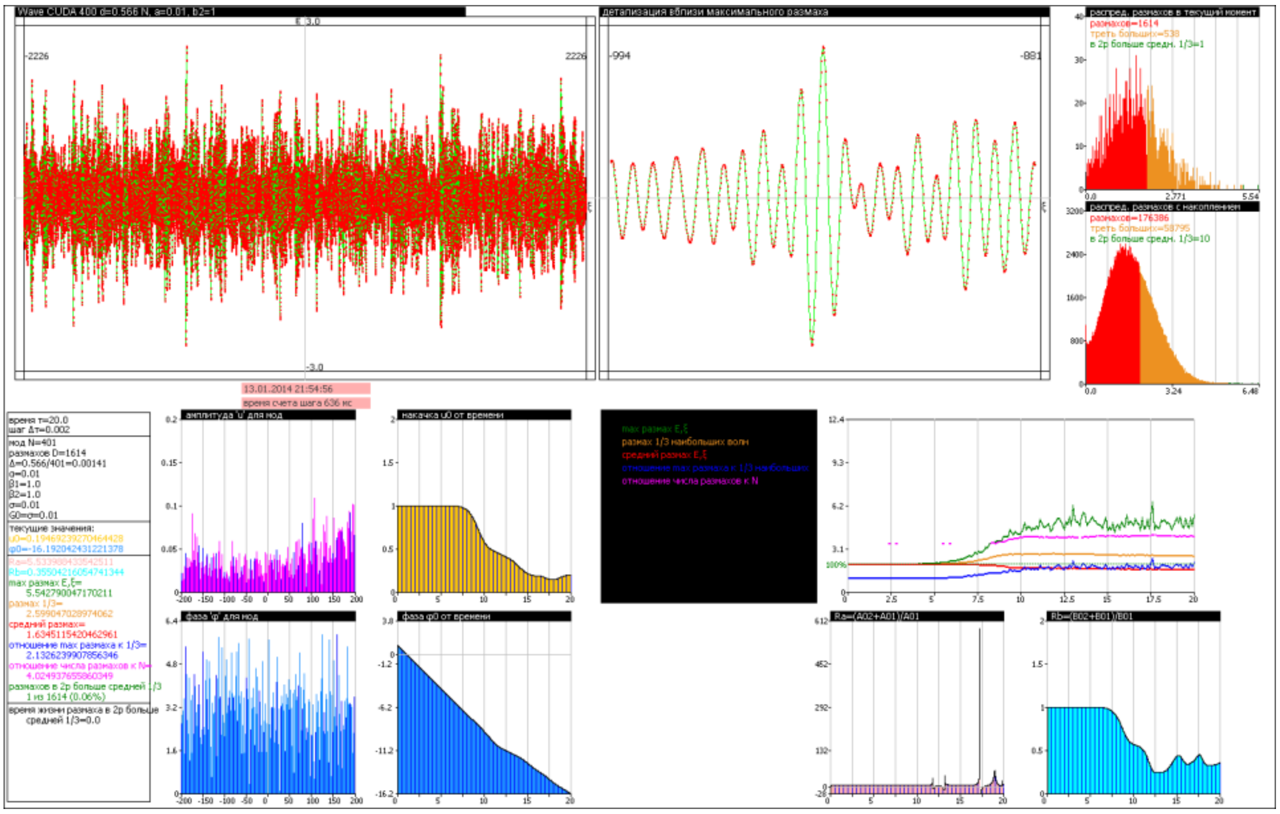

Wave CUDA 400 d=0.566 N, a=0.01, b2=1
детализация вблизи максимального размаха
распред. размахов в текущий момент
распред. размахов с накоплением
13.01.2014 21:54:56
амплитуда 'u' для мод
накачка u0 от времени
фаза 'φ' для мод
фаза φ0 от времени
Ra=(A02+A01)/A01
Rb=(B02+B01)/B01

Fig. 2. A diagnostic stand for analyzing the behavior of individual wave packets and statistical processing of the results of modeling the behavior of waves of anomalous amplitude in the ocean. Source [8,14]

Stages of forming a complete scientific theory. The first stage is the formulation of the problem. It should be noted that it is the critical mass of researchers concerned with solving problems within the framework of this problem that provokes interest in the scientific community and forms the fashion for these activities.

The second stage is an attempt to collect, summarize and discuss the material found and obtained. The community formulates questions, finds answers and participates in solving many individual problems, explains the details of the phenomenon or process under study.

The third stage – In order for the found data, facts and knowledge to acquire a complete form of theories, their synthesis is necessary. A paradigm of the theory being created is formulated, which synthesizes all the main results of research on this problem. The main task is to find a place for the theory being created in the general structure of scientific knowledge.

Let us consider the ten−year development of these stages of research using the example of the studied process of structure formation in the Proctor−Sivashinsky model of convection of a thin layer of liquid or gas [10]. 1. First, a seed paper was published, which presented the appearance of two spatial structures in a layer heated from below [11]. The nature of their appearance and replacement was unclear.

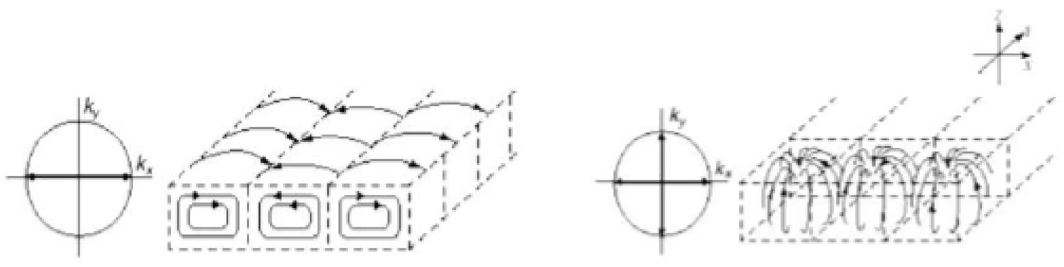


Fig. 3 Convective structures: rolls and square cells. Source [14]

2. Then, the previous calculations were checked and all collected data were structured [12,13].

3. Next, a set of specific problems was formed, the purpose of which was to find dependencies and correlations of some quantities in relation to others (see, for example, [14]), where the state function was identified that describes the change in the spatial structure (at a constant viscosity of the studied structure formation and under conditions of viscosity dependence on temperature). In addition, it was noted that when the topology of the structures changed, domains of a new phase appeared, and the number of boundary defects quickly decreased, simultaneously with the narrowing of the spatial spectra of the formed structure. The times of structural transitions increased, and the intervals of change in the state function decreased, in accordance
with the theory of relaxation.

To speed up the calculation and reduce errors, parallel computing methods were used [14]. Ultimately, all the signs of first− and second−order phase transitions were discovered and analytical and graphical evidence of their identity with general ideas about phase transitions was carried out.

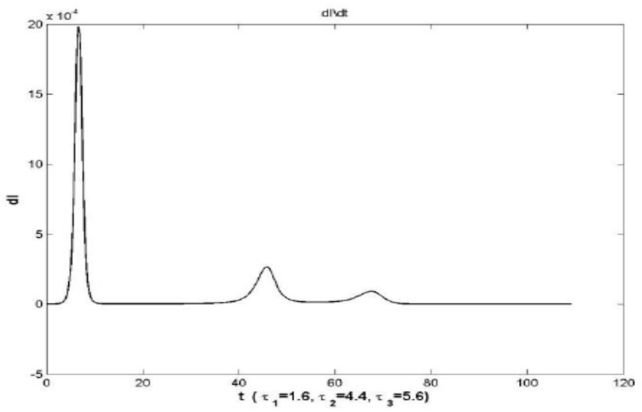


Fig. 4. The behavior of a state function derivative as a function of time. The characteristic times of transient processes are the time of occurrence of the “amorphous” state, the time of formation of pronounced shaft−like structures , and the time of forming a system of cells for one of the implementations of the process of establishing convective motion. Source [12-14]

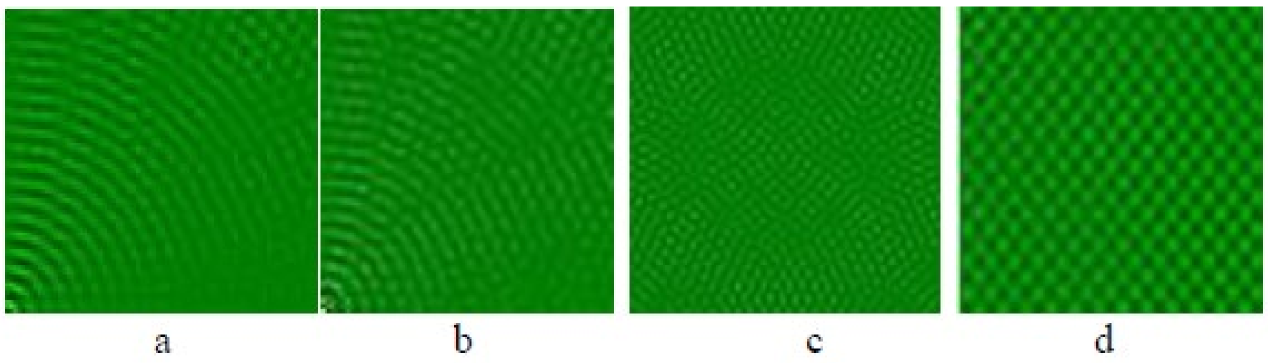


Fig 5. The appearance of fragments of the spatial structure of the temperature field distribution on the surface of layer: a) after the structural phase transition of the first kind with the formation of convective rolls, b) with the transverse modulation of the rolls, c) during the formation of domains – a metastable spatial structure, after the destruction of the roll system, d) with the formation of a stable convective structure – square convective cells. Source [12-14]

4. The synthesis of all these important details of the description and the final form of the completed theory of phase transitions of the first and second kind in the convection of thin layers of liquid or gas were presented in the monograph [8].

The results, diagnostic methods and technologies for solving a number of such large–scale problems are presented in the works of the Department of Artificial Intelligence, which existed for two decades at the V. N. Karazin Kharkiv National University, thanks to the support of the rector, academician V. S. Bakirov [15].

Simulation modeling of reality. But in many cases such reliable equations verified by theory and confirmed by practice do not yet exist. Then they use equations that, generally speaking, are not strictly derived from first principles. Such modeling is often called simulation. Such models based on intuitive ideas, at first glance, satisfy the phenomena under study. In this case, free parameters, due to their selection, would seem to be able to help satisfactorily describe a specific process. This is not a description of the phenomenon in fact, but an imitation, because the parameters are the conditions under which the process is realized. Changing the parameters changes the nature of the process: by selecting these parameters, it is possible to achieve similarity between the real situation and its description using the simulation model. Such an approach may be useful for practitioners who limit themselves to modeling this specific process in strictly defined conditions. Although there are doubts here, because with a noticeable change in parameters, the description of the phenomenon and process may differ from reality, and then the simulation model is able to let its designers down.

7.2. Logical–symbol problem solving systems

Logical–symbolic problem solving systems are based on direct and reverse deduction. A direct production system is built from the initial to the target state. It is necessary to identify states and goals, then it is possible to define a global database (i.e. a database and a knowledge base) for describing these states. Then direct rules are rules that are applied to descriptions of states to generate new states. This approach is useful if the number of target states is unknown, and the initial (original) state is one. In the case of a reverse production system, a set of descriptions of the problem goals can also serve as a global database. In addition, the initial states are given – here these are axioms. Moving from the goals (they can even be considered questions), the reverse rules – rules applied to descriptions of goals to generate subgoals – lead to the initial data − axioms. The movement from the target state to the initial one allows us to consistently find subtarget states − lemmas (the path from which to the target state is then shortened). This approach is rational to use if the target state is one, and there are many initial states. It is not difficult to see in this process the proof of theorems. The direct production system starts from the initial data. The database is expanded and the terminal condition can be a given number of steps (regardless of the result) or its complete filling (all facts from the initial set are included in the database). The importance of the terminal condition or the stopping condition is extremely important, otherwise the computing system will waste resources on endless calculations. The reverse production system starts its path from goals to the initial data − axioms. The coordination of the entire system of steps corresponds to the probable result. The question of the uniqueness of the solution remains (a standard question when proving theorems, since there can be several initial states).

Is it worth going back? The mode without return is suitable only if it is known for sure that the solution (search for a solution) method has been tested and will most likely lead to the correct result (which, generally speaking, is unknown to anyone), i.e. to the solution. It is useful to have a special function that implements the strategy of choosing rules. Sometimes the calculation mode with return is necessary. The calculation procedure must complete its work if the generated database satisfies the terminal condition. The list of facts and rules used to create the final database is output at the end. By the way, the recursive procedure does not have a stop (but it can be assumed by introducing restrictions on the recursion depth), it can generate new databases. In such modes, the system usually does not remember unsuccessful paths and many transformed (as a result of applying the rules) databases, although it can be made to remember all the steps of the last path and the databases that arise at these steps. A similar situation will arise if the robot creates a preliminary plan and then has to check and adjust it.

The beginning of the formation of logic (IV−III centuries BC). Socrates created an educational environment, conditions for the training of young Athenians and formed a famous group of researchers in the gymnasium at the temple of Apollo Lycia, but no materials about him have survived. However, his students, including Plato, wrote a lot about him. Relatives of the young men who studied and listened to Socrates, seeing that their children despised the archaic values inherent in their environment, decided to execute the teacher.

Plato (Aristocles), after the voluntary execution of Socrates, created his own school outside the city, already in some sense a fee−based one − the Academy. He left behind thousands of dialogues and other materials in the form of conversations, where characters argued and expressed their thoughts. The most famous works are "The State" and "Laws", written in the form of a utopia. He was the first of the earthlings to systematically study the interaction of the human mind with the world (which became the main theme of philosophy), developed the meaning of ideas in reflections, which was the first attempt to raise this issue, which later manifested itself in Neoplatonism and other philosophical movements. After Plato's death, his student Aristotle created the Lyceum school at the temple, where Socrates taught the youth. He changed the approaches to science and took up various topics in physics, geography, geology. But his main achievement, interesting for us, was the development of logic. After the death of his student Alexander the Great, he was forced to leave Athens and died in exile. Another of his students took the materials of Aristotle's research to Asia Minor, from where, two hundred years later, they came to Rome, where they were published. Aristotle's logic was studied for almost two millennia and gradually acquired a modern, more convenient form. This form in its current form can be presented in a formalized form, that is, in the form of mathematical logic.

Methods of logical constructions.1. Semantic method – it is necessary to show that under all interpretations the given statement is true. 2. Syntactic method – they look for a complete formal description based on the accepted formalism. Axioms are premises, initial data, the result of applying the rules is the conclusion. Axiomatic schemes and rules can be used.

An algorithm, as a scheme of reasoning and decisions, allows us to answer the question of whether a given sequence of steps proves that the statement is a consequence of the given axioms.

Application of predicate calculus – (1) all database expressions are converted into sentences in a certain normal form; (2) the calculation management system, the algorithm is built on the basis of the application of operations. In particular, in predicate theory, a developed form of mathematical logic, a resolution is used instead of an implication. In this theory, the negation of the goal is taken and a contradiction is sought, that is, a proof of the theorem by contradiction is used (this method was used by the ancient Greeks). (3) the system of productions in predicate theory is commutative and allows the use of a non−reversible control mode (where the order of application does not matter, and repetitions are not dangerous, the problem is only in reducing the solution time).

Strategies for selecting the applied sentences = predicates are built on the formation of a selection tree − an auxiliary graph; simplification of expressions occurs due to (1) the exclusion of tautologies and other expressions that are obviously true; (2) when evaluating individual literals (parts of a sentence) due to attached procedures (if a literal receives a true − T assessment, then such a sentence is excluded, and if false − F, then the sentence can be left, but the true literal can be excluded from it); (3) identifying the inclusion of one of the sentences in the composition of another with a certain substitution of variables (i.e. the first sentence to the substitution is a more general form of the corresponding part of the second), then this part of the second sentence can be excluded, which does not affect the unsatisfactoriness of the rest.

Constructive proofs (constructive theorems) − here it is not enough to prove the existence of a solution in the domain of definition of variables, it is necessary to find specific values of these variables (the solution itself). Non−constructive proofs indicate only the possibility of the existence of a solution.

Graph theory is a section of discrete mathematics. This is one of the branches of discrete topology. It refers to the theory of networks, to topological forms of description, network models of representation of any process or system. A graph is a geometric scheme that is a system of lines connecting given nodes. L. Euler is considered the founder of graph theory. In 1736, in one of his letters, he formulated and proposed a solution to the problem of seven bridges, which became one of the classic problems of graph theory. As a science, graph theory was formed after the appearance of the first monograph on this topic by D. Koenig (1936) [16]. A conceptual graph can describe expressions of predicate logic. Logical formulas are phrases of metalanguage. Predicate arguments are attributes, events, states. Predicate names indicate the method of connection (grammar rules, connection rules, procedures) between concepts. For example: graphs can be used to describe predicate theory, but the possibility of information loss should be taken into account.

Graph search is the formation of a set of rules from their total number and the databases generated by these applications to form a search tree. At the root of the tree is a description of the initial configuration. Rules − arcs lead to successor (descendant) nodes. Here, nodes correspond to the database, and arcs − to rules. A tree is a special case of a graph, each node of which has no more than one parent. If two nodes are simultaneously successors of each other, a pair of arcs is replaced by an edge. A node that has no parents is a root node, and one that has no successors is a terminal node. The root node has a zero depth, the depth of any other node is equal to the depth of the predecessor plus one. A sequence of nodes, each subsequent one of which is a predecessor of the previous one, is called a path, with a length equal to the number of nodes in the sequence. The control strategy for graph search can also be considered as a process of discovering a part of the implicit graph containing the target node. The tree (or rather, the bush) grows until one of its branches satisfies the terminal condition, and generally speaking, such a strategy is quite ineffective. Thus, unlike the previous case (i.e., the mode with return), the system remembers all previous paths, individual steps of each path, and modified databases formed as a result of these steps. The advantage of such a situation is that the system can start not from the initial position, but from the modified database that turned out to be closest to the result of the task, if the search procedure has criteria for the proximity of the intermediate state to the resulting state or, as in the PROLOG language [17], if further movement along the graph is difficult. On the other hand, it is useful to find conditions (generally speaking, heuristic ones) that still limit the number of branches, making the tree (bush) narrower.

Uninformed procedures are procedures that do not have heuristic information for ordering nodes and evaluating arc characteristics during implicit graph formation processes. For uninformed procedures, although they are rarely used in artificial intelligence systems, there are methods: depth−first search with depth constraints is performed each time to the maximum depth, then returns to the highest branching point and again to the very depth, etc. Heuristic methods are heuristic search methods − using additional information to reduce the search. They are based on minimizing the combination of the cost of the path to the goal (the sum of the values characterizing the arc elements on the path to the goal) and the cost (volume of calculations) of searching for this path. For this purpose, the so−called evaluation functions are introduced, based on which some nodes of the search tree can be thrown out of consideration. Step−by−step searches are searches " with a chosen floating point". They are used in the case of limited resources or large graphs and for cases of heuristic search. The information of the first stage is remembered and the second stage starts from the most acceptable (from the standpoint of heuristics, the optimal value of the evaluation function) vertex reached at the first stage.

Quality characteristics of work – search direction $P = L/T$, where L is the length of the path to the goal, T is the total number of generated vertices. Branching efficiency indicator – the average number of successors of a separate tree vertex.

In logical inference systems, generally speaking, when solving a whole series of problems, a multitude of graphs can also be formed (under certain conditions and a certain architecture), which are capricious logical connections of facts and rules. At first glance, it is difficult to expect that this system will allow creating associative connections between remote junctions. At the very least, this can weaken the requirements for the unambiguity of the solution, which mathematicians do not like. But if we consider the situation of brainstorming, searching for answers, then such a connection would be useful, strengthening the associations between solutions. If in a neural network connections arise in its very structure, then in an expert system such connections can be ensured by the joint use of facts and rules from a global database to solve a number of problems. Therefore, there is no need to be afraid to set a large number of tasks for an intelligent system, perhaps this will just enhance its creative capabilities. As with people, by forcing a person to solve many problems, you can see how his intelligence grows, usually assessed by the ability to identify unexpected solutions, which is called creativity. The PROLOG language, based on mathematical logic, practically on predicate theory, was actually created by Alain Colmerauer [17], who was a professor at the University of Montreal (where he created the Q−system) and then worked at the University of Marseille. PROLOG undoes the results of a part of the computation that led to failure and returns to the point where this unsuccessful branch began. When a procedure returns to a certain point, all the variable instantiations made after this point are cancelled. This order ensures that the PROLOG system systematically checks all possible alternative computation paths until a path leading to success is found, or until it turns out that all paths lead to failure. In essence, PROLOG tries to prove a theorem of the truth of a goal or, more precisely, an answer to a question based on the initial data.

Another language based on mathematics is the use of list processing technology, developed on the basis of Church's theory[18] (and developed according to its own logic), allows you to build a set (space) of solutions and, using comparison functions, select the desired answer. Create a set of functions that evaluate the level of coincidence of the desired (or declared a priori) goal with the base of facts and rules. This has some analogy with systems using logical inference technology, where you can search for a goal (direct deduction) or identify the correspondence of the goal to facts and rules, that is, prove a theorem (reverse deduction). A feature of functional languages is the ability to modify the description of the output, according to the ideas of the developers. But the resulting program can also answer questions that play the role of a goal here, like logical inference languages. That is why LISP and its analogs − functional languages − are not in vain considered the languages of artificial intelligence. In the extended lambda calculus, which is the basis of functional programming languages, in addition to the unnamed functions defined by lambda expressions, external constants are used: integers, symbols and logical values, a constant denoting the empty list NIL, as well as the designation of functions: "levels" and other operations on logical values.

The disadvantage of expert systems of logical programming was that they operated with two values "true" and "false". Individual facts, rules, goals − everything had a so−called true assessment on a two−point scale. In life, this does not happen, so they looked for an opportunity to move to a structure of fuzzy concepts closer to reality. Very opportunely, the theory of fuzzy sets and fuzzy logic was proposed to humanity by Professor Lotfi Zadeh [19.]of the University of California (1965). Fuzzy logic is located between formal and natural description; describes − approximates any mathematical model (which is proven by the FAT theorem [20]− Fuzzy Approximation Theorem, B. Kosko, 1993); but often the original data set is not only incomplete, but also slightly contradictory, the input and output variables entered by experts may turn out to be not quite adequately describing reality; therefore, adaptability is required from systems, i.e. they should allow correction of knowledge and parameters in the process of solving problems [21]. The creation of systems capable of coping with variable and incorrect problems is very complex. To solve variable and incorrect problems, large amounts of information are needed, even poorly structured, for example, a kludge − a confusion of images and connections between them. It is only necessary to conscientiously fill global knowledge bases with such information. Perhaps it is worth relaxing the restrictions on the level of incompatibility of substantive information and rules.

One should also beware of the information noise of the network structure. It is clear that the acceleration and increase in the volume of information flows will become a big problem for information technologies and intelligent systems. If we expect the feedback to become more complex, then we should strive to create mechanisms in the artificial intelligent development environment that are similar to human capabilities. This is the need to conduct an imaginary experiment and procedures for synthesizing embedded and received knowledge, the ability to find analogs in the existing knowledge of civilization. We must ensure that such internal imagination, when the system uses its own information resources as initial data, is able to actively develop its inner world, creating new knowledge, synthesizing it and looking for inaccuracies and inconsistencies. There is no need to be afraid that the system will sometimes "think". These are the tasks that are currently being actively considered. In addition, the discovered inconsistencies can give a reason for thinking to a person, such a clumsy creator of intelligent systems. All of the above prompted the use of neural networks

Semantic networks. C. Peirce proposed a prototype of a semantic network − an existential graph (1909). Later, this approach in the form of graphs and semantic relations was used by psychologists and physiologists. But computer semantic networks were developed by R. Richens (1956) for the purposes of machine translation. The meaning of semantic networks was in the development and diversity of connections between graph elements. Below we will discuss some features of the description of semantic networks. However, semantic networks contain, in addition to the logical connections between concepts inherent in graphs, something more important. A semantic network is built from the general to the specific, i.e. there are hierarchical structures that allow you to move from more general concepts to specific concepts. Therefore, the structure of semantic networks will always differ from a traditional graph by the presence of generalizations in it, allowing you to return to them when solving specific problems and find connections between, at first glance, different concepts. And the role of inheritance of properties, in particular, here becomes an important tool for finding answers.

The Semantic Web is the Internet where information is pre−processed by machines. The traditional Internet is based on HTML pages, information is retrieved using a browser by the user himself.

The Semantic Web uses the capabilities of the semantic network, processing information with a client program and providing the user with the result of logical processing. The term "semantic web" was first introduced by T. J. Berners−Lee and was defined by him as "the next step in the development of the World Wide Web." The lowest level is the Universal Resource Identifier (URI), a unified identifier that determines the way to write the address of an arbitrary resource. The Semantic Web, naming each concept simply using the URI identifier, allows everyone to use the necessary concepts. The next level is the XML language as a basic form of markup and tools designed to define and describe classes of XML documents (DTD, XML schemas).

An additional level is focused on working with a digital signature to determine the degree of reliability of data. The RDF layer allows searching for necessary concepts in the semantic Web. Based on XML, RDF resource description tools and RDF schemas are deployed to coordinate XML data on the network and create catalogs and dictionaries of concepts.

The OWL network ontology language allows organizing a more complete automatic processing of network content than supported by XML and RDF, providing additional semantic support along with formal semantics. An ontology defines the semantics of a specific subject area and helps establish links between the values of its elements. Ontologies are used to improve the accuracy of Internet searches – the search engine will only return sites where the desired concept is mentioned with precision, and not positions where a given keyword was encountered. The lowest level is the Universal Resource Identifier (URI), a unified identifier that determines the method of recording the address of an arbitrary resource. XML documents (DTD, XML schemas). An additional level is focused on working with a digital signature to determine the degree of data reliability. OWL ontologies allow organizing a more complete automatic processing of network content than that supported by XML and RDF, providing additional semantic support along with formal semantics. the sought–after concept, and not the positions where the specified keyword was encountered. The Semantic Web will initially be created in individual areas of science, technology, services, etc. Competitors of the Semantic Internet will be neural networks. The creation of specialized semantic networks will be slow, but industries will not refuse this. At first, specialized semantic networks will be created in the most formalized branches of science – computer technologies; here there will be no need to look for specialists of different profiles. Then it will be the turn of mathematics, physics, genetics and other branches with a high level of formalization, where symbolic transformations have already been developed. The creation of libraries and ontology languages adapted for such formalized areas of activity will not be a very difficult matter, the basis is already there. The main task will be the development of agent programs for processing texts and conversations. After this, the stage of synthesis of individual specialized semantic networks into a single network of semantic VEB will come, which will be built up slowly. Therefore, developers must be trained specialists in individual branches who know programming and understand information technology.

All the discussed systems of accumulation, storage and use of knowledge, as well as human mind, contain their own world, where all operations, types and actions are laid down earlier. However, the problem is not the creation of a universal format for presenting this knowledge, but the possibility of an adequate and universally recognized unambiguous presentation of it. It is extremely difficult for people to agree on a single presentation of knowledge not because it is impossible in principle. But first of all, because people do not see the need for it, because they can exchange data (even in a universal format) without using an unambiguous presentation (there is no need for unique solutions, mathematical rigor and unambiguity are not required). The associativity of information manifestation, multi–channel selection of images and carrying out procedures in various intelligent systems (it is proposed to use this term for logical inference systems to a greater extent than the definition of "artificial intelligence") is very different, but it occurs everywhere. There is a large share of heuristic elements here. Although this is "rather a necessity than a virtue." But it should be kept in mind that all systems created and being created form a world that is primitive to one degree or another. And methods of obtaining conclusions, decisions and recognitions based on deduction also have a limited scope of application. People and the devices they create solve problems either by rejecting excessively complex constructions or by constructing cumbersome description schemes populated by new entities and prejudices based on expected answers. Attempts to create theories "about everything" do not cease, demonstrating the excessive ambition and self–confidence of humanity.

7.3. Expert and recommendation (consulting) systems

Bayesian system of logical inference. The nature of expert systems can be discussed using the Bayesian model as an example. Thomas Bayes, an English mathematician, presented an extremely useful model of an expert system

Fig. 6. Thomas Bayes (1702−1761). Source [21]

His achievements have been widely used in the field of probability theory and mathematical statistics. He formulated the theorem bearing his name (1763), and the formula bearing his name allows one to estimate the probability of events using empirical data; the generally accepted terminology is: "Bayesian estimate", "Bayesian network". When using a Bayesian inference system, the information processed by the expert system is not absolutely accurate, but is of a probabilistic nature. The user does not necessarily have to be confident in the absolute truth or falsity of the evidence; he can respond to queries with some degree of confidence. In turn, the system provides the results of the consultation in the form of probabilities of the occurrence of consequences (see, for example, [22,23]).

1. The conditional probability p (A | B) is equal to the Ratio of the combined the probability of the event B, provided that it is not equal to 0.

$p(A|B) * p(B) = p(A\cap B)$ and similarly $p(B|A) * p(A) = p(B\cap A)$

since $p(A\cap B) = p(B\cap A)$, then we get the ratio

$\mathbf{p(A|B) * p(B)= p(B|A) * p(A)}$ ––– Bayes theorem

2. For non–intersecting events $B \cap A$ and $B \cap \neg A$, $B = (B \cap A) \cup (B \cap \neg A)$ holds, then

$$p(B)=p((B\cap A)\cup(B\cap\neg A))=p(B\cap A)+p(B\cap\neg A)=p(B|A)*p(A)+p(B|\neg A)*p(\neg A).$$

$$\textbf{P posterior} = Py * P / (Py * P + Pn * (1 – P))$$

P posterior = p(A|B) − the probability result A, if there is event (symptom) B

Py = p(B|A) − the probability of event (symptom) B, if there is the result A,

Pn = p(B|−A) − the probability of event (symptom) B, if there is not the result of A,

P − a priori probability result A

(1−P) − a priori probability of absence result A

Theory of Dempster − Shafer. The measure of truth is determined by the interval

trust < measure of truth < plausibility

Arthur Pentland Dempster (1929) – a mathematician, professor at Harvard University,

Fig.7. Arthur Pentland Dempster (right), Glen Shafer. Source [21]

Glenn Shafer – a professor at many universities, a mathematician in the field of statistics and artificial intelligence. They developed methods for assessing the reliability of events, in particular, the Dempster−Shafer theory (DST).

Credibility is the sum of the masses of evidence supporting a hypothesis. Plausibility = 1 − the sum

of the masses of evidence rejecting the hypothesis. Let us consider an example. It can be assumed

that a girl has one person that she likes or does not like.

Table 1. TRUST < MEASURE OF TRUTH < PLAUSIBILITY Source [21]

Hypothesis	Weight of evidence	Trust	Plausibility
Indifferently. That is, she does not have a friend	0	0	0
She loves someone	0,2	0,2	0,5
She has one, but she doesn't love him	0,5	0,5	0,8
Universal. She has a friend, but it is not known whether she loves him or not	0,3	1,0	1,0

7.4. Algorithms and computational graphs based on mathematical logic

The most well−known version of mathematical logic is predicate theory. The mathematicians who created it sought to minimize the number of symbols and parameters in order to o.btain the simplest description system. As is often the case, and we will encounter this in other cases discussed below, this approach is not always convenient in practical applications. However, the standard set of symbols needed for the formal description of sentences of the language of mathematical logic is quite small in the more general case.

∧ - *conjunction – AND* *	∃ *of existence*
∨ − *disjunction − OR* * ,+	*(* ∃ *x − such x is found*
⇒ −− *implication − IF ... THEN* **	∀ *community*
−− *objection − NO*	*(* ∀ *x − for each x)*
⇔ *equivalence*	

* ∨ – *is a possible disjunction in the strict sense of what it divides*

** *IF − antecedent, premise, TO − consequent, conclusion.*

Fig. 8. Relationships- quantifiers of mathematical logic. Source [21]

In predicate theory, there are two forms of description: conjunctive and disjunctive. In the conjunctive form, it is sufficient to use disjunction and negation, assuming that conjunction connects all sentences of the language by default. In the disjunctive form, only conjunction and negation were left, and disjunction connects all sentences by default. It is clear that the conjunctive form has received the greatest distribution, since the connection of all sentences with the help of conjunction is the most understandable and convenient. In addition, since implication, with the help of which it was possible to connect individual parts of sentences (literals), was excluded, a new operation was needed − resolution, which actually replaces it. The resolution of two sentences is formed by forced disjunction of two mutually opposite literals P and ~ P that are in two different sentences (correctly constructed formulas − CCF).

Procedure: For example, let us take（R ∨ P) and (~ P ∨ Q), take the disjunction of these statements R ∨ P ∨ ~P ∨ Q, (which, generally speaking, does not contradict each of them) or, which is the same R ∨ (P ∨ ~P） ∨ Q, then we will simplify this new expression, removing the tautology（P ∨ ~ P） and finally get the *resolvent* − that is R ∨ Q. This turned out to be a new statement that can be entered into the global database.

Explanation: The resolution of the two statements P ∨ Q and ~P ∨ G is Q ∨ G. On the other hand, these statements can be represented as Q ∨ P and P ⇒G with P replaced by G and we get Q $\vee$ G. And that is just to avoid using implication!

It makes sense to immediately demonstrate the possibilities of transforming expressions: *Tautology* = identically true = A $\vee \sim$ A and contradiction = identically *false* = A $\wedge$ $\sim$ A. As shown by F. L. G. Frege NOT F1 $\vee$ F2 has the same truth value as F1 $\Rightarrow$ F2.

Fig.9. Friedrich Ludwig Gottlob Frege Source [21]

The contributions to logic come from the contributions of Aristotle and K. Gödel. Frege (1879) created the logic of predicates, the introduction of quantifiers, and much more, which led to the book of Principia Mathematica and Gödel's non–uniformity theorem [24]

In addition, set theory implies *de Morgan's rules* $\sim$ (x $\vee$ y) = $\sim$ x $\wedge$ $\sim$ y; $\sim$ (x $\wedge$ y) = $\sim$ x $\vee$ $\sim$ y. Used as a rule, first–order predicates, where arguments cannot be other predicates. In the composition of two clauses with variables, additional literals that are possible under some substitution (for example, x = y) are distinguished (that is, P (y) and ~ P (x) and transformed into P (x) and ~ P (x) , then only within this substitution it is possible to apply the resolution rule and obtain a resolvent.

Task: if Zhora goes to the same places where Kolya goes, and Kolya is at school, then where is Zhora?

$\forall(x)\,[B(KOLIA,\ x) \Rightarrow B(GORA,\ x)]\,;\ B(KOLIA, SCHOOL)$

The main question of the task "Where is Zhora?" can be answered if there is a solution, in other words $(\exists x)\,B(GORA,\ x))$, it is a consequence of the initial knowledge base (axiom).

Here $(\exists x)\ B(GORA,\ x))$ is the target. The denial of the target is $(\forall x)\,[\sim B(GORA,\ x)]$. So let us find out if there is a solution. It is important to note that even without asking a question, one can find that the answer can be obtained from the axioms themselves. Indeed $-B(KOLIA,\ y) \vee B(GORA,\ y)$, with and $B(KOLIA,$ $SCHOOL)$ let us obtain $B(GORA, SCHOOL)$ with appropriate unification.
Let's build a decision tree, that is, a solution algorithm.

Table2. Refutation based on the resolution. Source [21]

	$\sim B(KOLIA, y) \vee B(GORA, y)$ axiom 1	$B(KOLIA, SCHOOL)$ axiom 2
$\sim B(GORA, x)$ Refutation which is the goal	$\sim B(KOLIA,\ x)$	*NIL*

Why should the result be an empty set here? Let us take the negation of the question $\sim A\ (x)$ and make a disjunction with $A\ (a)$ vailable in the database. When $x = a$, we get a tautology, which after simplification (its elimination) will give an empty sentence. But often exactly $A\ (a)$ is not in the database, but it can be obtained using the resolution of sentences from the database. Then this is an example of a non–constructive theorem, because it speaks of the existence of the solution, but does not provide a way (technology) of a solution and does not show it.

Although it is possible to get a clear answer, but for this, instead of negating the target sentence, one should take a tautology based on it $\sim B(GORA, x) \vee B(GORA, x)$.

Predicate Theory on Graphs. Expression $Q(v, A) \wedge [[\sim R(v) \wedge \sim P(v)] \vee \sim S(A, v)]$, *Let's present it in the form of a AND/OR graph* $Q(w,A) \wedge [[\sim R(v) \wedge \sim P(v)] \vee \sim S(A,v)]$. The scheme of constructing a graph (often such a graph is classified as a hypergraph) is simple: conjunction is represented in the form of traditional connections, and disjunction in the form of k−connections, i.e. united by a link. Note that such a representation of a hypergraph is usually characteristic of direct production. The set of sentences into which such a compound phrase can be transformed (of course, conjunctively connected to each other) is found on the final vertices of the graph, $Q(w,A)$; $\sim R(v) \vee \sim S(A,v)$; $\sim P(v) \vee \sim S(A,v)$. After applying the rule to a given expression − fact, $\sim S(x,y) \Rightarrow R(x) \wedge Q(y)$, we can obtain a new set of solutions. Substituting the rule into the assertion graph and applying unifiers to the variables

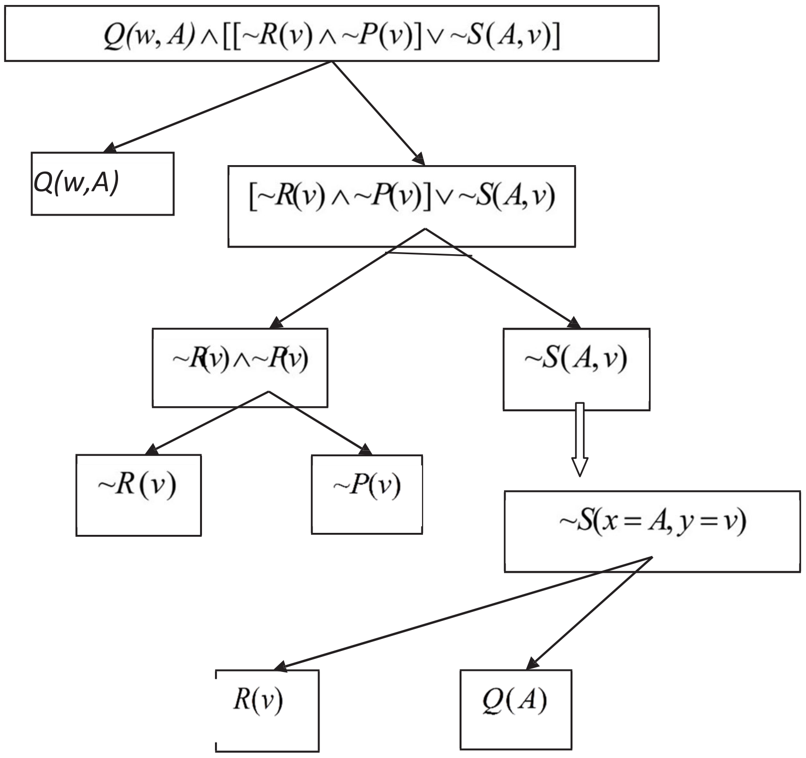


Fig.10. Joining a rule to a graph of facts (direct system of products). Source [22]

This is a graph of direct production, from a set of axioms that were included in the compound sentence and after applying the rules, we obtain a set of new solutions ~P(v) ∨ Q(A); ~P(v) ∨ R(v); ~R(v) ∨ Q(A); and the resulting tautology ~R(v) ∨ R(v) can be excluded.

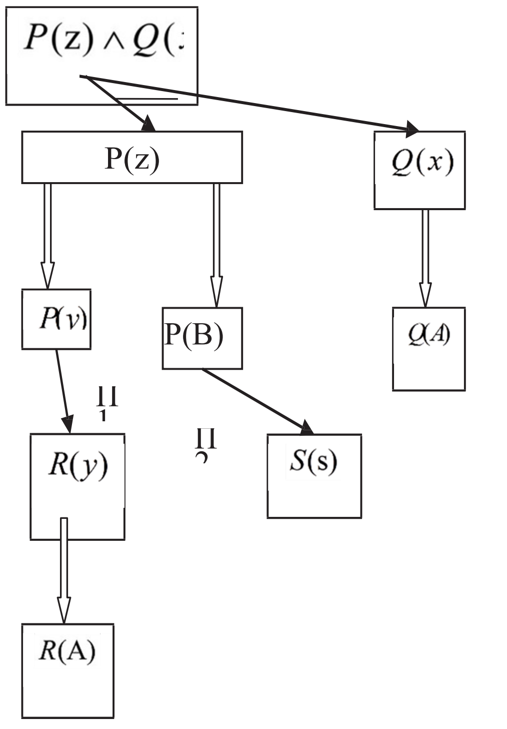
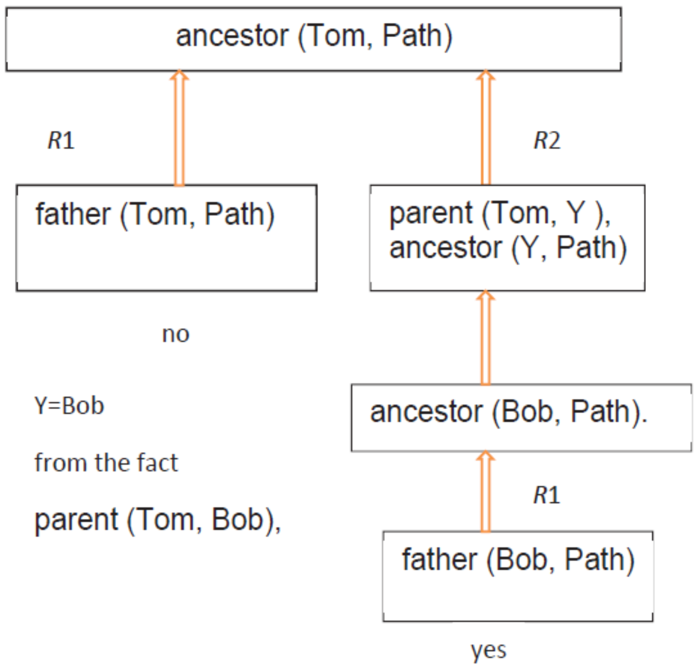
Backward production. PROLOG ***language.*** Let us prove the goal $P(z) \wedge Q(x)$, the facts are $R(A)$ and $Q(A)$, and the rules are as it follows: R1: $R(y) \Rightarrow P(y)$ and R2: $S(z) \Rightarrow P(B)$.

In fact $P(z) \wedge Q(x)$, the combined goal is that each literal is also a goal and these two subgoals $P(z)$ and $Q(x)$ both must be reached. The rules here are easier to apply in the form of implication. That is, the rules are applied in the reverse order, for example, to the literal $P(z)$, which is the first subgoal, we shall apply the reversed rule $P(y) \Leftarrow R(y)$ under the condition $z = y$, then the new subgoal $R(z)$ and $z = A$ is achieved, since was a fact in the database. On the right branch of the graph, the subgoal $Q(x)$ is similarly achieved by substitution $x = A$, since there is a fact $Q(A)$. It is interesting that the branch of the graph built on the rule P2 forming a new goal $P(B)$ is not achieved in any way, but since it does not affect the joint decision, this branch is simply discarded. In other words, R2 is not necessary to solve this rule.

In the PROLOG, reverse products are used. The creation of a new sentence from the goal G (A) and the rule P (y) G (y). Here, the head of the rule is G (y), and the body of the rule (conditional part) is P(y). When the goal G(A) coincides with the body and head of the rule G (y = A), both are removed and the body of the rule P(y = A) remains with the replaced variable (concretization). This is now a new goal. The above is equivalent to the procedure (look Fig b)

Demonstration of LISP work. Recently, from the group of LISP languages, Common LISP, which replaced the Interlisp language, has attracted more practical interest. Common LISP, along with PROLOG, has long been the best artificial intelligence language. By the way, pure LISP included a set of functions that allowed creating and modifying lists consisting of sub–list elements. By adding a new element (sublist) to the beginning of the list or removing the main part of the list or its remainder, and using the conditional transition operator, it was possible to program on it.

1. In modern LISP, procedures can become data, substituted into expressions as arguments.

Let us prove the goal , the facts are $R(A)$ and $Q(A)$, and the rules are as it follows: R1: $R(y)\Rightarrow P(y)$ and R2: $S(z)\Rightarrow P(B)$. The left partial graph is the substitution A = y = z. The average partial graph is the substitution B / z. The right partial graph is the substitution A / x.	*head* *body* **Facts** father (Tom, Bob). father (Bob, Path). **Rules** R1. ancestor (X, Z) :– parent (X, Z). R2. ancestor (X, Z) :– parent (X, Y), ancestor (Y, Z). **Question** (goal) ? – ancestor (Tom, Path).
 $P(z)\wedge Q($ P(z) $Q(x)$ $P(v)$ P(B) $Q(A)$ $R(y)$ $S(s)$ R(A) *a*	 ancestor (Tom, Path) R1 R2 father (Tom, Path) parent (Tom, Y), ancestor (Y, Path) no Y=Bob from the fact parent (Tom, Bob), ancestor (Bob, Path). R1 father (Bob, Path) yes *b*
Fig.11 a.Exclusion of the branch of the graph with the inverse system of products. Source [21]	**Fig.11 b. Task graph (reverse output). How the PROLOG solves Source [21]**

2. At first it was thought that LISP is a language only for recursive schemes, but

in practice

it turned out to be quite flexible.

3. LISP allows you to formulate and remember arbitrary sentences (idioms) that are useful

for artificial intelligence systems.

4. It is possible to write programs other than AI on it, for example, AUTOCAD is written in it.

Task: Alyosha, Vitya and Igor found a small kettlebell on the floor of the physics room after lessons. Each of them, examining the find, made two assumptions. Alyosha said: "This kettlebell is made of brass, and it weighs, most likely, 5 g" (L5). Vitya assumed that the kettlebell was made of copper and weighs 3 g. (M3) Igor believed that the kettlebell was not made of brass and its weight was 4 g. (M4) The physics teacher was glad that the missing item had been found, and told the guys that each of them was only half right. What metal – brass (L) or copper (M) – is the kettlebell made of, and what is its weight? In the answer, write down a list that includes the first letter of the name of the metal, and then the number corresponding to the weight of the kettlebell, for example (L 4).

Compare var: (M3), (L3), (M4), (L4), (M5), (L5);   Initial data (L5) (M3) (M4).

```
defun cross(a b)
  apply (function append)
      mapcar (lambda (x) mapcar (lambda (y)(listx y))b))a)))
      ⇨  (M3),(L3),(M4),(L4),(M5),(L5),
defun check (v a)
(+(?(car v)(car a))(?(cadr v)(cadr a))))
defun is(var)
cond((and(=1 check var'(L5)))
             =1 (check var'(M3)))
             =1(check var"(M4))))var)))
  Defun solve(list)
         (cond
                  ((nul list)  nil)
                 ((is(car lst)))
(t(solve(cdr lst)))))
solve(cross’(m,l’(3,4,5)))
==> (M5)
```

We will write such sets where the elements coincide, at least in one list

Initial data (L5) (M3) (M4).

1 group (L5) includes	(L3),	(L4),	(M5)
2 group (M3) includes	(L3)	(M4)	(M5)
3 group (M4) belong to it		(L4)	(M5)

Thus, the solution is a distribution of coincidences, the maximum number of coincidences gives the correct answer, but the presence of a distribution of coincidences leaves a feeling of uncertainty, that is, there is still no clear answer.

Fuzzy logic algorithms. The theory of fuzzy sets and fuzzy logic, created in the early sixties of the last century by the professor of the University of California Lotfi Zadeh [19], became the first attempt to move from logic, where there is truth and falsehood, to more vague definitions. The apparatus of fuzzy logic describes − approximates any mathematical model (this is evidenced by the theorem FAT − Fuzzy Approximation Theorem, B. Kosko, 1993). In addition, the so−called characteristic function was introduced − the function of belonging of an element to some set, − takes a value from zero to one. Each object in this case was related to a number of sets, which expanded the boundaries of applicability of this description.

Tsukamoto's algorithm**.** With the help of experts, vagueness is introduced in the form of fuzzy sets = membership functions A_1 (x), A_2(x), B_1(y), B_2(y), but here C_1(z) and C_2(z) are clear, defined functions that allow you to find clear values of z according to each of the rules R_1 and R_2;

R_1: if $x \in A_1$ and $y \in B_1$, to $z \in C_1$; R_2: if $x \in A_2$ i $y \in B_2$, to $z \in C_2$.

find truth levels−cutoffs

$$\alpha_1 = A_1 (x_0) \wedge B_1 (y_0), \qquad \alpha_2 = A_2 (x_0) \wedge B_2 (y_0) ;$$

from the equations $\alpha_1 = C_1\ (z_1)$ and $\alpha_2 = C_2\ (z_2)$ let us find clear values (answers regarding the application of individual rules) $z_1,\ z_2$;

4) the answer is determined, for example, as a weighted average.

Thus, the initial knowledge is presented in the form of sentences using conjunction (they are used at the stage of obtaining the cut−off levels) and implication. In this case, implications may not be transformed and used directly (at this stage, the values of the characteristic functions of fuzzy sets C are determined). Further, since all sentences (for the set of sentences where conjunction is used) are connected by conjunction, it is used to obtain the final answer in an indefinite form. The transition to a clear form can be different, in particular, as suggested in the traditional examples (algorithms) given. Generally speaking, the transition to a clear form is defined incorrectly in fuzzy logic algorithms and networks if there are no expressed extrema of the integral (calculation result) membership function. This means that the solution is not sufficiently defined and requires clarification of the algorithm structure.

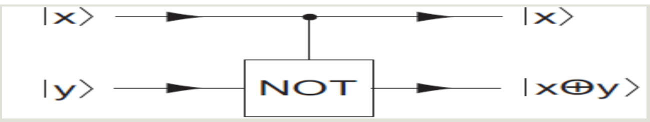
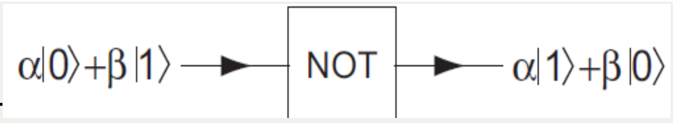
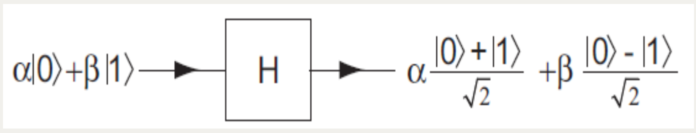
Conjunction and disjunction (on the right) of fuzzy logic could be represented as three possible variants

Table 3 - Variants of conjunction − disjunction pairs of fuzzy logic Source [22]

$\min(\mu_A, \mu_B)$	$\max(\mu_A, \mu_B)$
$\mu_A \cdot \mu_B$	$\mu_A + \mu_B - \mu_A \cdot \mu_B;$
$\max(0, \mu_A + \mu_B - 1)$	$\min(1, \mu_A + \mu_B)$

Formal logical descriptions in quantum computing. On the parallelism of quantum computing you can use all available N q−bits (therefore, it is better to have many of them) as substitutes for the full basis. The scheme calculates the value of f(x). At the same time, in the internal space of the quantum computer, all or many values of this function are obtained.

The output to the real space during measurement will give only one value of the function, but without leaving the quantum space, you can continue to calculate (calculate other properties of this function).

logical elements	classic	quantum (must be reversible) act only on ground states
NOT	0 <NOT> 1, 1<NOT>0 one−bit	one−bit 
Hadamard gate	no analogue	one−bit 
AND conjunction	0,0<AND>0, 1,1<AND>1 0,1<AND>0 1,0<AND>0,	no analogue (since the classical gate is irreversible)
OR disjunction	0,0<OR>0, 0,1<OR>1 1,0<OR>1, 1,1<OR>1	no analogue
XOR (NOT, if top is 0 (1) second bit does not changes	0,0<XOR>0, 1,1<XOR>0 0,1<XOR>1 1,0<XOR>1,	two−qubit CNOT gate (quantum XOR) 

The simplest fuzzy neural network. Let's first build an algorithm: Rule 1: if x ϵ A_1, then y = z_1 and rule 2: if x ϵ A_2, then y = z_2.

Fig.12. Logical operations in classical and quantum computing

Source: http://www.theory.caltech.edu/ preskill/ph229

$$A_1(x) = \{1 + \exp[-b(x-a)]\}^{-1}, \quad A_2(x) = \{1 + \exp[b(x-a)]\}^{-1},$$

and the degrees of truth

$$\alpha_1 = A_1 \, , \quad \alpha_2 = A_2$$

they are often called cutoff levels. The transition to clarity (defuzzification) can have the form of a weighted average

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}$$

This was the algorithm, and now we will consider that this is a learning network [25], where A_1 and A_2 are activation functions – in essence, these are neurons in fact. In the learning process, the parameters can be adjusted as follows, based on the type of the error function

$$a := a - \eta \cdot \partial E(a,b) / \partial a \, , \quad b := b - \eta \cdot \partial E(a,b) / \partial b$$

you can choose the error function E (a,b) as the sum of the squares of the difference between the current value and the true one, if it is known.

Fuzzy neurons. Let us define the following operations : T = min(μ_A , μ_B); S = max (μ_A , μ_B). We shall define neuron "I" as y = T (p1, p2) = T (S (w1, x1), S (w2, x2)), where p1 = S (w1, x1). We also shall define the "OR" neuron as y = S (p1, p2) = S (T (w1, x1), T (w2, x2)), where p1 = T (w1, x1).

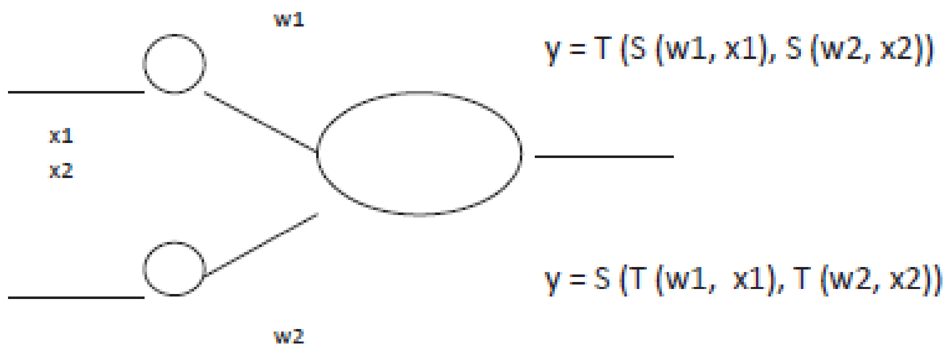

w1
x1
x2
w2
y = T (S (w1, x1), S (w2, x2))
y = S (T (w1, x1), T (w2, x2))

Fig.13. Fuzzy neurons. Source: [22]

The work of neuron I can be illustrated as it follows. First, the signals of a large number of agents are selected, which exceed the threshold (it can be different for each agent) of their perception by the neuron. This threshold is set by the synaptic weight. From those signals that exceeded the threshold and from the background of other synaptic weights, the minimum value is chosen, which is reaction of a neuron. It is similar to how the power structure chooses its minimal reaction to the demands of active people from the crowd.

Neuron OR is, on the contrary, the selection of the maximum value from the set of signals that do not exceed the permissible threshold (determined for each agent by the synaptic weight) and the background of other synaptic weights. Here it is similar to how the authorities choose the most effective solution from all proposals that do not violate some restrictions, for example, they reject overly radical and unacceptable proposals.

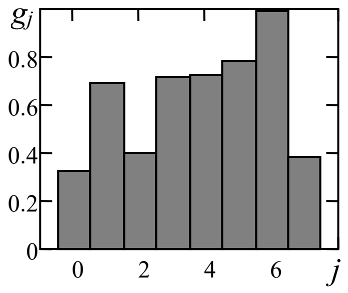
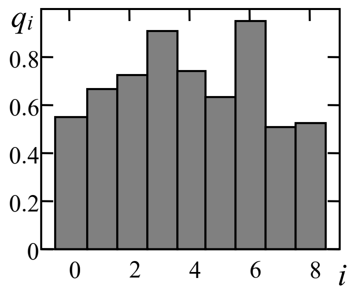
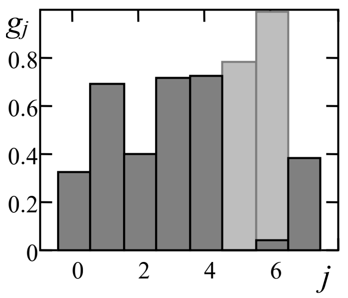
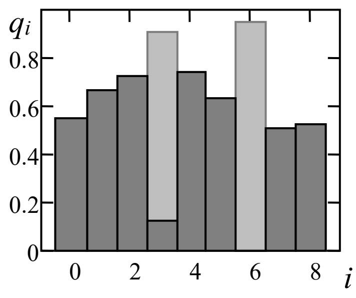
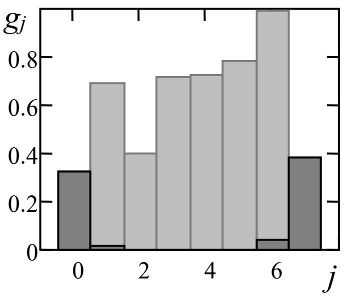
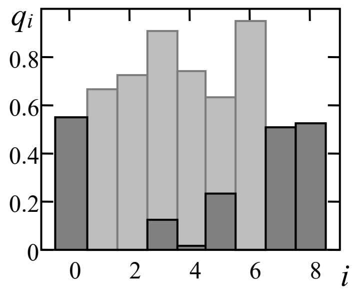
Model of exchange trading. Let sellers want to get the price $y_j = y_{j0} - g_j \frac{1}{1-G} f_y,$ buyers are satisfied with the price $x_i = x_{i0} - q_i \frac{1}{1-G} f_{x,}$. The difference between prices is the spread, which should be reduced to zero to start trading. Here g_j and q_i are coefficients that correspond to the intentions of the participants. Strengthening the motivation to participate in trading is determined by the value 1/(1−G), where G is the share of the sold product, f_y and f_x are random variables that vary from 0 to 1, the trading conditions can be written as $T[(y_1, y_2, ..., y_N)] = S[(x_1, x_2, ..., x_M)],$ где $T[(y_1, y_2, ..., y_N)] \equiv T[(y_1 \wedge y_2 \wedge ... \wedge y_N)],$ $S[(x_1, x_2, ..., x_M)] \equiv S[(x_1 \vee x_2 \vee ... \vee x_M)],$ where what actually means if min(y_j)> max(x_i), trading stops. Let's assume that all the goods will not be sold and bought during the session. Modeling of this process by my graduate student Aleshina M. led to the following results, shown in the diagrams.

Problem solving by a robot. the task of creating plans within the STRIPS (Stanford Research Institute Problem Solver). Task description [22].

The state of the world is described: predicates CLEAR(x) − the upper surface of cube x is not occupied, ON (x,y) − cube x is on cube y, ONTABLE (x) − cube x is on the table. The state of the robot's hand: the predicates HANDEMPTY − the hand is empty, HOLDING (x) − the hand holds the cube x. The actions: **pickup** − take from the table, **putdown** − put on the table, **stuck** −put on another cube, **unstuck** − remove from another cube.

Table.4. Implementation of trades in one session

Source: diplom graduate student Aleshina M.

Number of trades, k	Sellers	Buyers
k=0.		
k=2		
k=6		

The description of this state: CLEAR (B); CLEAR (C); ON (C, A); HANDEMPTY; ONTABLE (A); ONTABLE (B) − cubes A and B lie on the table, cube C is on cube A. The description of objectives: ON (B, C) ∧ ON (A, B) − cube A is on cube B, and cube B, in turn, is on cube C.

Action model: taking a cube from the table

pickup (x)

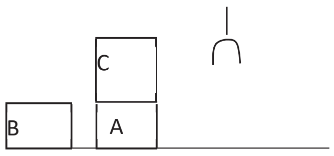
P (precondition): ONTABLE (x) ∧ HANDEMPTY ∧ CLEAR (x).

D (delete list): ONTABLE (x), HANDEMPTY, CLEAR (x).

A (add formula): HOLDING (x).

Matching substitution in this case gives − pickup (x) can be applied only if x = B. Then the new state description is: CLEAR (C), ON (C, A), ONTABLE (A), HOLDIBG (B).

TASK:Let us consider the situation when cube C stands on cube A, the surface of cube B is free, and cubes A and B are on the table. The hand is free. HANDEMPTY, CLEAR (B), CLEAR (C), ONTABLE (B), ONTABLE (A), ON (C, A)



Let's build a strategy selection sequence (plan):

{ **unstauk** (C,A), **stuck**(C,B), pickup(A), **stuck** (A,C)}

Table THE FORM OF RECORDING PROCEDURES Source[21]

HANDEMPTY ∧ CLEAR(C) ∧ ON (C, A) ∧ CLEAR (B)
unstuck (C, A)
HOLDING (C) CLEAR (A) CLEAR (B) HOLDING (C) ∧ CLEAR (B) ∧ CLEAR (A)
stuck(C, B)
ON (C, B) ON (A, C) ON (C, B) ∧ ON (A, C)

The STRIPS system behaves simply. Having received a task, it solves part of it (a subgoal), and if difficulties arise with solving the next part of the task (for example, another subgoal), it is able to destroy the previous solution without thinking and achieve the second subgoal. And then, as if nothing had happened, return to the first subgoal. This resembles the behavior of a person who immediately gets down to business, without thinking about his actions, but acting according to the situation. At the same time, it is able, if it does not work out right away, to start working again and again until it achieves success. A more intelligent RSTRIPS system, the behavior of which resembles the behavior of a person who, before acting, develops a detailed plan of his behavior. Moreover, if something in the plan does not work out, he adjusts this plan, revises it and strives for its (the plan's) rationality and consistency. So that later, in the process of its implementation, he does not drive himself into a corner and does not create conflict situations [22].

7.5. Connectivist systems

First, W. McCulloch and W. Pitts created the first model of a neuron (1943), then in 1960−1970, crude models of neural networks appeared. However, it all started with the perceptron of F. Rosenblatt (Frank Rosenblatt) MARK1, which already in 1960, confusedly, recognized some letters of the alphabet.

But few people paid attention to the development of dramatic events provoked by the Pontifical Academy of Sciences. In 1964, the school "Brain and Conscious Experience" under the auspices of the Pontifical Academy of Sciences and under the leadership of J. Eccles, who was famous for having discovered the chemical side of information transmission in synapses. But at the same time, he and many of his colleagues continued to see some other supernatural nature in the structures of the mind and memory, although they recognized that experience is formed in the patterns of neurons and through the activation of synapses. The meeting was unable to answer J. Eccles' question of how exactly neural activity in the cerebral cortex provokes sensory experience. The reaction of G.−L. Teuber was more categorical: "When we tried to outline the probable systems and mechanisms without which there would be no consciousness, the opinions expressed were very different and contradictory. There was not even certainty ... regarding the understanding of why consciousness is needed" (see [26]). Nevertheless, having freed himself from the influence of religiosity and having gone through the realization of his mistakes, R. Sperry later formulated the conclusions after this meeting that formed the basis of the cognitive revolution: the process of consciousness is impossible without the brain, and mental factors that are consistent with the phenomena of subjective experience, but do not imply disembodied supernatural forces acting outside the brain mechanism, are also associated with it [Sperry J.C. Perception in the Absense of theNeocortical [27]. Useful were the considerations of J. Eccle and the philosopher K. Popper that thought is capable of influencing the state of the brain, and T. Nagel that this state can correspond to external influence [28]. For neuropsychologists, the opinion of J. von Neumann that the nervous system functions in conditions of parallelism was also important, especially since the structure of the brain, as it turned out, consists of neural clusters oriented to different functions with strong internal short connections and weak connections between themselves [29,30]. Thus, all nervous processes, including thought processes, were finally attributed to the activity of the brain, although how exactly this was implemented in the array of neurons remained unclear.

Since most scientists were religious people, it was not strange for them that the human brain and its nervous system provide control over all organs, but thoughts, as they imagined, are connected with the spiritual, immaterial sphere. Therefore, up until the middle of the 20th century, despite the enormous amount of information about the human body and, in particular, about the brain, even serious neurobiologists and neurophysiologists did not part with this idea. Since the spirit was responsible for the thought process, even the nature of the appearance of intelligent beings on the planet (the emergence of the second signal system) was not discussed, perhaps the widespread idea of the origin of the world, instilled since childhood, was to blame. A detailed study of the brain was needed, the involvement of mathematicians in thinking on this topic, and, most importantly, the first attempts to create connectivist models of the brain from artificial neurons were needed to dispel the ethereal fog of spirituality of thinking. Indeed, a small number of neurons in the model turned out to be able to find answers, that is, generate a thought. There was no longer any room for the ethereal spirit. This made an impression. People who had sobered up from the religious intoxication began the process of rapidly creating and mastering models of the brain − neural networks.

As usual, from a complete denial of the possibility of describing intelligence by systems with real elements (here these are neurons with synapses) they quickly moved on to the intention of creating a thinking machine that would be no inferior to the brain of a thinking person.

As is customary in the scientific community, a decade later, F. Rosenblatt's former classmate M. Minsky convincingly criticized the capabilities of the perceptron, which led to a delay in the development of neurocomputers for another decade. M. Minsky specified the memory sizes for weight coefficients and pointed out the problems with too much memory for large problems and showed that the training and solution times could be too long, which caused extreme pessimism among developers. In addition, M. Minsky has a saying about the properties of a network: “if a neural network is randomly connected, this does not deprive it of a pre–established idea of how to play, it is just that the game will be unknown to you”. That is, what is happening in the network is unlikely to be clear to its creator. Although people still decided to use the perceptron for its intended purpose, because nothing else came to their mind during this time. A neural network is a web of neurons that, in order for it to do something meaningful, must be trained beforehand. Give it several tasks with answers and adjust the network parameters so that it demonstrates these answers, all from this training set. It is desirable that there are more tasks, then there is a chance that a new task will be solved as needed. David Rumelhard and Jeffrey Hinton (1986) proposed the most common method of network training today – the backpropagation method: having estimated the degree of deviation from the desired answer, they recommended going through the layers of the network in reverse order and correcting the settings. And to this day, all neural networks, neurocomputers, which number millions of libraries, are trained according to this scheme, although there are networks that do not require training.

The convergence theorem of a perceptron trained by the error correction method, proved by F. Rosenblatt and colleagues, states that for any initial weights and sequence of procedures, this method will always lead to a solution in a finite period of time. It was also shown that with too large a training volume, neurons may not be enough, and their number must be increased. If after training the tests show that the errors are decreasing, then it means that you are lucky, you have learned. And if the errors on the test examples are increasing, then the network simply remembered everything that was told to it during training, and did not really learn anything, this is politically correct to call overfitting. Overfitting of large networks is expected, because an increase in free control parameters is equivalent to an increase in the degrees of freedom. An insufficient number of training tasks allows to ensure the correctness of the required answers only to these tasks. However, other similar tasks may be solved incorrectly. In addition, it is impossible to ensure a direct correspondence of the initial conditions to the final results. Different initial conditions may give the same or similar results. It is difficult to combat overfitting, but some approaches have proven successful. They began to look for softer forms of neuron activation, complication of connections of neurons of different layers (for example, ResNet architecture and similar ones). But training has its own problems − a large number of correctly solved problems − examples are required (for this purpose, teams of developers of such examples were even created), with a large number of layers the effect of corrective corrections is weakened, in addition, it is necessary to carry out many tuning operations − in fact in the iteration mode. It turned out, for example, to be useful to pre−train the network for processing data of a certain type − the method of pre−training the network by D. Hinton (2006). A promising procedure is the use of parallel computing when tuning the network with CUDA technologies on graphics cards (GPU − graphics processing unit). Already for cellular communication, special processors − NPU (neural processing unit), so−called artificial intelligence accelerators (AI accelerator) have been created to increase the performance of parallel computing similarly to GPU with noticeable energy savings.

Often used to weaken the influence of neighboring neurons on each other (for quantitative criteria of such weakening, the well−known methods of Tikhonov regularization in mathematics are used), use disconnection during training of large sections of the network − the original training method of D. Hinton Dropout (2012).This temporarily reduces the complexity of the network, creating conditions for suppressing overtraining, and then when connecting previously disconnected sections, their adaptation to the new state improves. In addition, the human brain has found a way to fight overtraining (albeit not as successfully as mathematicians would like), disconnecting entire sections of the cerebral cortex that are not needed to solve a simple task.

But if all this does not help, then the network was considered incapable and abandoned. Although there is another reason for the appearance of unexpected answers. When the network is powerful enough, it is able to get out of the class of solutions imposed on it by the training program and offer its own, somewhat different, which the experimenters did not expect. For them, the network disrupts the established picture of the world. Such network intelligence was also not welcomed, and this overly intelligent network was also refused. Although a more rational solution could be a permanent increase in the number and quality (commonality, diversity) of educational tasks to preserve the stability of the world picture. In the author's book "On the Benefits of Reflection", the hypothesis of "necessary development" was expressed.

Any evolving intellectual system (in particular, a civilization) is constantly increasing its intellectual computational resources (i.e., more and more new blocks of the intellectual network are connected), its complexity is growing. And the previous volume of training tasks and, accordingly, the proposed solutions soon turns out to be insufficient. In the sense that problems arise with the formation of derivative solutions, inconsistencies grow, which are perceived as errors. That is, the stability of the picture of the world is disrupted. The only way out here can be to increase the volume of known tasks that ensure effective training of the global information and computing system. This increase can be supported by science, an exit from the closed circle of previous concepts and ideas through observations and experiments.

The requirement of uniqueness of the solution, imposed by mathematicians, in the subsequent practice of creating artificial intelligence for solving non–computable problems that do not require uniqueness of the solution, based on neural networks of a huge volume, turned out to be unnecessary. The fact is that a person himself sometimes, when solving a problem, discovers many possible ways to solve it and is not at all bothered by the lack of uniqueness of the solution in the so–called non–computable problems (where, in addition to everything, there is also no formalized method of solution and there is no verification of the result), he chooses from them the one that seemed acceptable to him.

However, neural networks remained a "black box." It was impossible to fully understand what was happening inside with the signals, much less the information. That is why J.−S. R. Chang from Taiwan University had such an impressive guess: to transform a fuzzy logic algorithm with fuzzy neurons into a neural network[16]. In this case, it was possible to follow all stages of solving the problem. For example, in three−layer networks, the first layer introduces fuzzification based on reliability functions (characteristic functions), the second introduces fuzzy rules, and the third leads to the real result (defuzzification). It is important to note that the sigmoid functions of neurons are completely similar to fuzzy logic operators. But this is only the first step to understanding the processes in the "black box." It is worth paying attention to the ambiguity inherent in neural networks based on fuzzy logic in the form of the resulting characteristic function, which is not eliminated in the defuzzification process. Thus, from the standpoint of crisp logic, the conclusion is completely inaccurate. The transition to actual results is not rigorous, and the meaning of a specific value of the characteristic function during the transition to defuzzification is more of a mystery than a technique.

There already exist ANFIS (adaptive network fuzzy inference system) − 1. at the input − values are transformed into characteristic functions, 2. T−norm is used, 3. normalization of values, 4. formation of a fuzzy template, 5. defuzzification. And there is also NNFLC (neural network control system based on fuzzy logic = controller) − a combination of a fuzzy controller and a neural network.

Later specialized networks appeared − recurrent and convolutional. The idea of the initial classical recurrent networks (RNN) was to step by step supplement the text with the most probable next word. At each step, the output h_t (in the form of a hidden description) is calculated, which depends on the inputs x_{t-1} x_{t-2}, ... x_{t-m} at the previous steps. Later, recurrent networks began to form sentences sequentially, but now they used an encoder and decoder (in fact, two networks), which, using dictionaries, initially create a low−quality translation between language A (encoder) and language B (decoder). Then, using word selection methods to increase the overall probability of the sentence, a higher−quality translation is formed. The further development of recurrent networks already used the Bahdanau attention mechanism [31] (see also https://d21.ai/ and [32]) initially in the Seq2seq method [33-35.]. Taking this approach into account, the "Transformer" technology was then created [36,37], where the attention mechanism allowed us to abandon the recurrent mechanism of forming phrases and corpora "from word to word" and exclude LSTM and GRU memory blocks, now the sentence was already considered all at once.

The use of hidden descriptions has also proven useful in modern models. The use of the Vision Transformer encoder and the Convolutional Neural Network (CNN) decoder to obtain hidden representations of individual sections of the analyzed real−space radiation and subsequent field reconstruction is discussed [38]. Similar field reconstruction can be carried out using diffusion models [39]. The inclusion of new data, simulating such mappings with a complex network was also discussed in [40].

This attention mechanism also came in handy for other specialized neural networks, in particular for convolutional networks. It is important that such language networks, which are already considered traditional, contain fixed and identical activation functions in the structure of each neuron (that is, they can be considered on the nodes−neurons of the graph of calculations). It was possible to go further and create networks with different activations (LAN), but still in the nodes−neurons.

In the work [41], the possibilities of detecting regularities, in particular, integrals of systems of differential equations describing conservation laws are demonstrated. To obtain new results [42], it is useful to use known theories and data in combination with the introduced conditions of the problem, but it is also possible to discover patterns without resorting to a priori knowledge (see, for example, the [SciNet model [43], although even modern neural models can often be useless without a priori clues and especially in the absence of interactive communication. But hopes for the possibilities of applying mathematics, in particular for the rational evolution of known solutions introduced into the initial conditions of differential equations [44], allow us to obtain a result. Although it is important to find new patterns. We can, as before, introduce data arrays, counting on discovering clusters in the data structure or other correlation relationships that allow us to check our guesses. Another way is to construct a further evolutionary development of the characteristic processes previously discovered in the initial studies and to form a complete picture of the phenomenon. But of particular interest are the methods of searching for correspondence between the found patterns and ideas in existing theories and descriptions of well−known phenomena. In the equations, we can adjust the parameters of the description, which is equivalent to asking questions, as well as customize the description of the presented data. This is the essence of machine learning.

The encoder generates hidden states that are fed to a neural ODE suitable for handling evolutionary equations and nonlinear dynamics. The sequence of processed hidden states is then reconstructed by the decoder and after regularization (KL divergence [43]), a solution can be obtained [45]. Typically, neural networks are used to solve ordinary differential equations (ODEs) [44]. Although neural ODEs do not require prior knowledge of the system to build dynamic models based on large datasets, they are large and do not allow for deep interpretation of the results or extraction of physical information, but are still useful. Supervision of Nonlinear Dynamics (SINDy) is also useful for similar classes of problems [46]. The SINDy approach uses sparse regression to select candidate features needed to detect trends present in datasets. Using a pre−defined set of candidate features provides SINDy models with significant interpretability. Physically informed neural networks (PINN) [47] use dynamical system equations to form a loss function to train a traditional neural network. The loss function is based on physics, allowing training that is both data−driven and fits the known equations used. These networks are highly interpretable due to such governing equations, which gives physical meaning to the trends and patterns learned. The authors of [48] attempted to combine a neural ODE suitable for solving ordinary and partial differential equations with a highly accurate and interpretable KAN network, where the ODE allows variable time and space steps depending on changes in the computational grid. This allowed KAN−ODE to converge with a small number of parameters

But the variant of Kolmogorov–Arnold networks (KAN) turned out to be more interesting for scientists. The emergence of Kolmogorov–Arnold networks, which have, generally speaking, different activation functions already on the synapse weights (i.e., on the edges–weights of the computation graph) [49,50,32] has greatly interested developers and has caused a large number of publications and searches for new variations of the architecture (see, for example, [50]).

KAN does not have linear weights in the traditional sense at all – each weight is replaced by a one–dimensional nonlinear function, in particular in the form of a spline, although it can be another. It is important that each such function has free parameters that can be adjusted in the same way as the weights of a traditional network were previously adjusted. It is possible to construct a matrix of functions multiplied by the coefficients being learned on each layer. The description of such a matrix is a special case of KAN with two layers. And the generalized KAN is an empirical composition of such matrices [51].

$$f(\vec{x}) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1,i_L,i_{L-1}} (\sum_{i_{L-2}=1}^{n_{l-2}} ...(\sum_{i_2=1}^{n_2} \phi_{2,i_3,i_2} (\sum_{i_1=1}^{n_1} \phi_{1,i_2,i_1} (\sum_{i_0=1}^{n_0} \phi_{0,i_1,i_0} (x_{i0}))))...) \qquad (1)$$

After such a decisive step, which, unfortunately, was not supported by a strict theorem (omission of mathematicians), it is possible to work with KAN as with ordinary networks: add and remove neurons, combine layers. Gradually, the threat of incorrect use of multilayer KANs in the minds of developers weakened, and in addition, new proposals appeared for modifying splines, rather their representation, which accelerated the obtaining of solutions. For example, the authors of [49]). replaced splines with a set of simpler functions, for example,

Ri(x) = [ReLU(ei − x) × ReLU(x − si)]2 × 16/(bi − ai) 4 (2)

which significantly accelerated training and calculations. However, other activation functions are already used [53-55.]. In a similar direction, the effective use of GPUs with ADAM technology (see Jerry–Master. Jerry–master/kan–benchmarking).

7.6. Using models

1. ***Using the PINN model*** to simulate the behavior of an oscillator. In this case, the differential equation describing the process is second order. Here is a commentary on the program.

```
import tensorflow as tf import numpy as np
omega = 1.0 # Own frequency. Parameters of the harmonic oscillator
u0 = 1.0 # Initial position. Initial conditions
v0 = 0.0 # Initial speed. Initial conditions
num_points = 100 # Generation of the training set (time points)
t = tf.linspace(0.0, 10.0, num_points)[:, tf.newaxis] # Time points
class PINN(tf.keras.Model): # Definition of a neural network
def __init__(self):
super(PINN, self).__init__()
self.hidden_layers = [tf.keras.layers.Dense(20, activation='tanh') for _
in range(3)]
self.output_layer = tf.keras.layers.Dense(1)
def call(self, t):
x = t
for layer in self.hidden_layers:
x = layer(x)
return self.output_layer(x)
model = PINN() # Initialize the model
@tf.function # Loss function
def loss_fn(t):
with tf.GradientTape(persistent=True) as tape:
tape.watch(t)
u_pred = model(t) # The predicted solution
u_t = tape.gradient(u_pred, t) # The first derivative
u_tt= tape.gradient(u_t, t) # The second derivative
f = u_tt + omega**2 * u_pred # Physical equation
loss_eq = tf.reduce_mean(tf.square(f)) # Losses for the physical equation
 # Losses for initial conditions
t0 = tf.constant([[0.0]]) # Initial moment of time # Losses for initial conditions
with tf.GradientTape() as ic_tape:
ic_tape.watch(t0)
u0_pred = model(t0)
u0_t = ic_tape.gradient(u0_pred, t0)
loss_ic = tf.square(u0_pred - u0) + tf.square(u0_t - v0) Calculation of initial
```

```
condition losses
return loss_eq + loss_ic
return loss_eq + loss_ic
optimizer = tf.keras.optimizers.Adam()# optimization
@tf.function
def train_step():
with tf.GradientTape() as tape:
loss = loss_fn(t)
gradients = tape.gradient(loss, model.trainable_variables)
optimizer.apply_gradients(zip(gradients, model.trainable_variables))
return loss
num_epochs = 5000# Training
for epoch in range(num_epochs):
loss = train_step()
if epoch % 500 == 0:
print(f" Epoch {epoch}: Losses = {loss.numpy()}")
import matplotlib.pyplot as plt # Check the result
true_solution = u0 * tf.cos(omega * t) + (v0 / omega) * tf.sin(omega * t)
# True solution
u_pred = model(t) # predicted solution
plt.figure(figsize=(8, 5))
plt.plot(t, true_solution, label=' True solution', linewidth=2)
plt.plot(t, u_pred, '--', label=' PINN solution', linewidth=2)
plt.xlabel('t')
plt.ylabel('u(t)')
plt.legend()
plt.title(' Harmonic Oscillator: PINN vs True Solution')
plt.show()
```

oscillator pinn.py http://localhost:7619/92b1f218-c843-43fe-8f41-30d8f770d1e4/

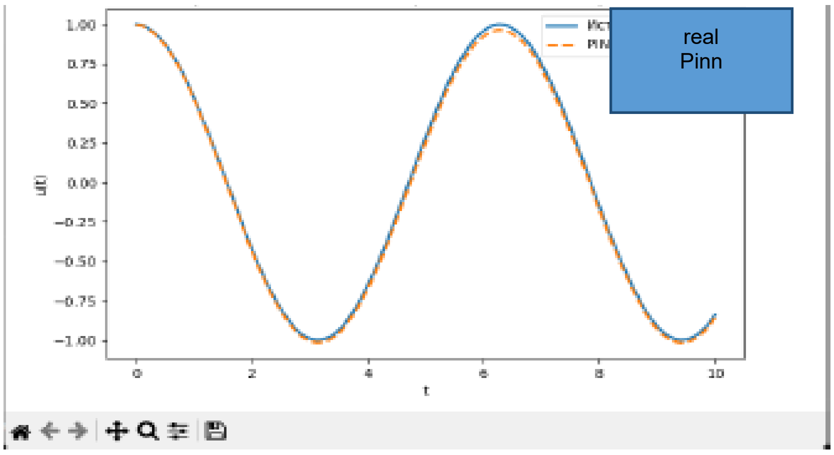


Fig. 14. PINN model for simulate the behavior of an oscillator. Source [autor]

2.KANN model - Kolmogorov-Arnold network. Creation and training. Since this is a new version of the network, which differs from the traditional one, we will make some comments. The parameters of the KAN network consist of external degrees of freedom (DOFs) - the number of nodes and links, and internal degrees of freedom in the G splines. In traditional networks, only external degrees of freedom exist. In KAN, the input variables are summed in the first layer. It is used

$$f(x_1, x_2, \dots x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p)\right) \quad (1)$$

Kolmogorov-Arnold theorem, where $\varphi_{q,p}(x_p)$ are one-dimensional ($1D$) functions of each input x_p that are being learned, and Φ_q is also a one-dimensional function into which everything is summed up.

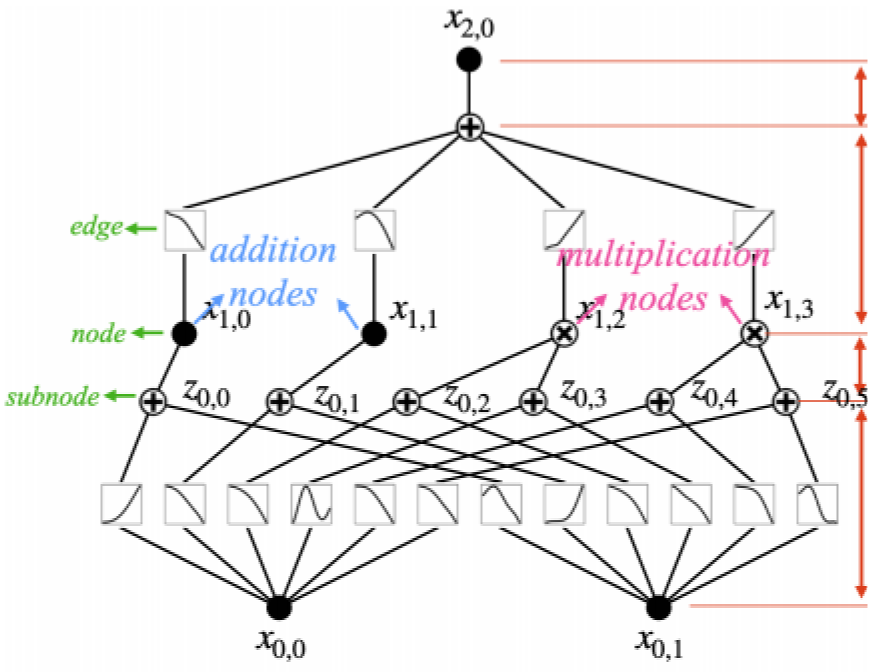


Fig. 15. Two-layer structure of the KAN network. Source: [49-51]

The network has nodes, subnodes and connections (see Fig. 15). Nodes are neurons that have a connection with functions that are learned in connections (edges). Subnodes are the outputs of functions that are then added or multiplied $\varphi_{i,j}$ (these are KAN functions). The transformation matrix-additions KANLayer Φ_l (with neurons that only add signals - addition nodes) and (multiplication layer) the multiplication matrix

M_l (with neurons that only multiply signals - multiplication nodes) takes the input vector $x_l \in \mathbb{R}^{n_l^a + n_l^m}$ and outputs $\vec{z}_l = \Phi_l(x) \in \mathbb{R}^{n_{l+1}^a + 2n_{l+1}^m}$ (multiplies the values of two nodes, so the coefficient 2 appears in the exponent). By the way, n_a here is the number of additive neuron nodes (addition nodes) and n_m - the number of multiplication nodes (multiplication nodes) in one layer. As for Fig.15, the widths of the layers that compose $n_a = [2,2,1]$ and multiply $n_m = [0,2,0]$. The *model.tree()* method generates a KAN structure, after training it forms a hierarchy of nodes (splines in the tree - architecture) how they are connected and which splines are used. The *plot_tree* method creates a visualization of this tree generated by the *model.tree()* method if pruning is done, so insignificant nodes can be darkened or discarded.

```
model=KAN(…)                          # creation and learning KAN
tree_structure=  model.tree()         # getting the tree structure
model.plot_tree ( style= ‘tree’)      # visualization in the form of a graph
```

If you need to create a network based on a mathematical formula or expression, for example - $f(x,y) = \sin(x) + y^2$ this means that *kanpiler* will automatically create a KAN network based on this formula, for example for regression problems and input data dependence analysis, which can be checked for compliance with this expression. The *kanpiler* function accepts variables as *simpy.Symbol* objects on the output *simpy.Expr*

.

```
from sympy import symbols, sin
from kanpiler import kanpiler
x, y = symbols(‘x y’)                 # Defining variables
expr =sin(x) +y**2                    # Output function
model = kanpiler([x, y], expr)              # Creating a KAN model
```

Moreover, KAN, without the help of the user, checks whether the data fits these expressions - equations (if the model is trained, but if not trained, then it adjusts to the data). Transition to new network variants using the *seed* label, which indicates different initializations of the model. It is important that if this is symbolic regression, then training is not required. The peculiarity of this type of network is that the data can change the initial symbolic expression, correct it and use other functions if you call the function:

```
model.train(dataset)            # data-based training
```

Sometimes two networks are created. The first is larger with sufficient and even redundant connections and nodes for training on the data. Splines are tuned to the data. No restrictions are made so that the network finds its own structure. Training is longer, there are many unnecessary ties. The network then thins out the connections itself, finds out which variables form independent modules. Why create a second network: The first finds dependencies, breaks it into modules, thins out the structure. The second uses ready-made modularity to optimize the structure. learns much faster, because all the restrictions are taken into account. In addition, it is easier to find symbolic functions in it.

If you do not enter a library of symbolic functions, then the network independently uses the *suggest symbolic* function to select candidates (activations) based on the data of symbolic activation functions - trigonometric, exponential, etc. This function can be called:

```
suggested_ activations=model.suggest_symbolic(input_data)   # call the suggest
                                                               symbolic function
for activation in suggested_ activations:                     # using activations
print(activation)                                           # candidate demonstration
```

If there are already symbolic functions (*kanpiler*), the network is compiled from equations-expressions, then it makes sense to immediately set sparse_*init= True*

model = KAN(…,sparse_ init= True).

If symbolic functions are not introduced, only data, then it is better to start with a dense network. However, the network is learning, but at first it preserves all connections.

Modularity is often distinguished - specific, intermediate, general - fundamental. At the beginning of creating a network, modularity does not exist, kanplier introduces symbolic functions and thereby suggests what tree can be made, and then when training

KAN, the tree is finally formed, but *plot_ tree* shows this.

After training, you can obtain the original expressions in the form of an analytical function or an

asymptotic expression.

```
import sympy as sp        # function input sympy
x, y = sp.symbols(^x, y*)                      # input variables
output = sp.sin(2*x+3*y)+sp.exp(x)- sp.log (y)      # комбінуємо декілька активацій
Taylor_expansion = sp.series(output, x, 0, 5)       # Taylor expansion at x→0
```