Article

Multiple-Valued Logic, Vocabularies Structure, and Linked List for Data Verification in Dialog Communications of Agents

Alexey Bykovsky *

Posted Date: 27 November 2024

doi: 10.20944/preprints202411.2173.v1

Keywords: data verification; multiple-valued logic; modelling of robotic agent; linked list; dialog communications

Article

Multiple-Valued Logic, Vocabularies Structure, and Linked List for Data Verification in Dialog Communications of Agents

Alexey Bykovsky

P.N. Lebedev Physical Institute RAS, Moscow, Russia 119991; bykovskiyay@lebedev.ru

Featured Application: The proposed method is aimed at the simplified integration of robotic agents into a local network, even if initially an agent possess only partially correct model and the verification procedure is to determine incompatible or missing components. The data verification and dialog data exchange scheme use the coded vocabularies structure of parameters and logic variables, supported by the logic communications language and the dialog phrase protocol.

Abstract: Distant verification of autonomous agent`s parameters in dialog mode is a difficult multi-parametric task, if the large-scale scene of action is characterized by a large number of collaborative and rival robots. The possible scheme to realize it for mass robots is to use non-exhaustive and selective data verification, combining the polling of internal subsystems and external data storages in collaborating network agents. Selective extraction of data for such checks is proposed to involve the special ordered set of vocabularies, which contains coded digital words and classifies parameters of agents, tasks, objects, and events. Such structure of vocabularies is to be combined with various versions of the linked list scheme, known in blockchain and actual for protective documenting of critical data. Multiple–valued logic is here the convenient method to provide autonomous navigation in a multi-parametric structure of data and verification variables.

Keywords: data verification; multiple-valued logic; modelling of robotic agent; linked list; dialog communications

---

1. Introduction

Modern research in civil robotics includes at least such fields, see **Figure 1**, as unmanned logistics and Internet of vehicles (IoV) [1], power grids with smart metering [2], "smart" city [3], autonomous robotic manufacturing [4], Internet of Things (IoT) [5], and medical nursing [6]. The general goal of these investigations, which are based on artificial intellect (AI) concepts of agents and multi-agent systems (MAS), is to imitate human abilities and to design autonomous and self-sustaining large-scale robotic systems, which can operate in wide space-and-time bands without permanent human control. However, the design of autonomous robotic agents is a complicated task, and the development of such systems involves the whole spectrum of conjugated technologies, including the research of modern mathematics [7,8], 5/6 G Internet [9], microprocessors [10], neural networks (NN) [11], fuzzy logic and controllers [12], computer vision [13,14] and speech procession [15], big data procession [16], traditional and post-quantum cryptography [17], quantum key distribution [18], and quantum computing [19]. The design, debugging, and modification of autonomous agents includes autonomous data verification and correction of errors and faults, caused by natural factors or cheaters [18].

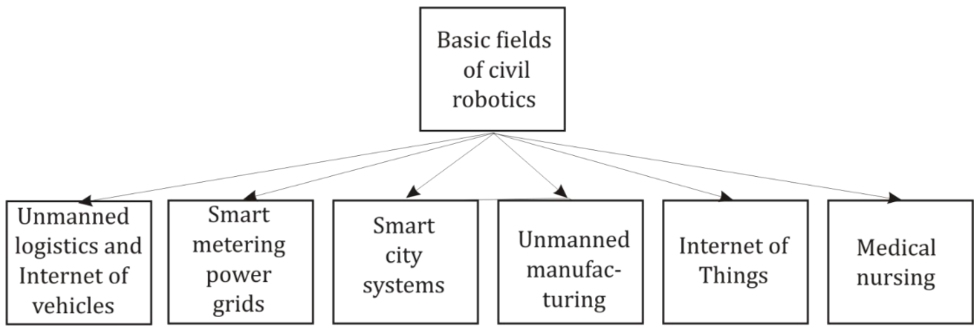

Basic fields of civil robotics
Unmanned logistics and Internet of vehicles
Smart metering power grids
Smart city systems
Unmanned manufac-turing
Internet of Things
Medical nursing

Figure 1. Basic fields of research for civil robotics.

The final goal of any type of verification procedures in agents is to exclude possible incidents and crashes caused by faults, attacks and illegal data modifications [20–22]. An ideal verification procedure should take into consideration all possible vulnerabilities of agent`s architecture, hardware, and software. It also should take into account current tasks, situation in the scene of action, the trust level of data sources, processed materials, characteristics of objects, and the workload. However, the problem also is to create and support the reliable referent (or template) model for data verification, as attacks can be aimed not only at agents but also at collective resources. Alternative way is to use verification with selective extraction of reference content from the distributed structure of local data storages, using free memory resources in loyal robots and network nodes. Respectively, verification procedure should compare data taken both from internal modules and external data storages, what needs to design the scene description formal model and to use protected from modifications distributed ledgers, partially reproducing the linked list (LL) scheme in blockchain [23,24]. These new options are to complement known data verification methods reviewed further.

1.1. Verification Tasks in Modern Robotics

In the review covering more than 150 papers published in 2008-2018, the analysis was done of formal specification and verification methods for individual and collective robotic systems. The specification was considered as the mathematical description of necessary actions for hardware and software components of the robotic system, which doesn`t determine the method of realization but can verify processes. There were considered many mathematical aspects of the problem, e.g., set–theoretic methods, logic methods of dynamic verification of hybrid systems, temporal logics, timed automata, robots communication languages, agent programming languages, finite state machines, state machines, Petri nets, probabilistic methods, Markov networks, automatic Buhi machines, model verification tools, and ontologies. The review has also covered models of physical environment, certification and trust indicators, external and internal threats. Finally, emphasis was made on the necessity to design integrated formal methods (iFM) for the specification of agent`s tasks, and the enlarging need was accented for the heterogeneous sets of formalisms for complicated systems description, where each one is responsible for its specific component. One more actual problem is the lack of methods to monitor the interrelation between the specification and the executed software code [8]. The authors of review has expressed the hope that the research in this field will help to create Integrated Verification Environment.

The practical realization of heterogeneous sets of formalisms concept and route verification method was demonstrated in for the cosmic robot Curiosity. Other publications has accented the set of actual aspects in formal verification schemes for robotic collectives; among them one should note the peculiarities of constructing both quantitative [26,27] and qualitative [28,29] models. Active research is being held in such fields as predicate systems modelling [26], temporal logic [29–31], and epistemological extensions [32]. Also one can see publications devoted to discrete relational models [33], state-based models [31,34], and beliefs-based ones [29]. Good results were obtained with the help of domain-dependent embedded models of Matlab/Simulink [35]. One should note the trend to design dynamic and non-exhaustive verification methods [8].

In the field of verification models for collective robotic systems, the detailed review has also emphasized the following problems:

- too general estimation parameters and the absence of methods to take into account the specifics of domain knowledge,
- the absence of realistic schemes to evaluate the applicability of selected criteria,
- the lack of specialized languages to model domain knowledge,
- insufficient development of methods for runtime verification.

Among other significant gaps, the authors of also has pointed out the lack of real possibilities for quantitative specification, i.e. of ability of known languages to describe the desired properties of the system. However, due to the inhomogeneity of agent`s structure, the languages for robots behavior description should provide as the discrete modelling of dynamic processes, as the continuous one. They are to model stochastic and epistemological characteristics of agents' interaction with the environment and users. The gap is too large between quantity indicators for intermediate and final tests results. Some other publications also discussed the need for methods to analyze probabilistic and stochastic processes [36], discrete and continuous dynamic processes in agent systems [38]. In paper the emphasis was placed on the fact that namely system description languages should track combinations of stochastic and continuous processes. Very substantial drawback is the lack of public technologies for checks and tests [27,34,40], and complicated systems with the detailed structure description now has no appropriate tools for the model-based testing [35,41] and the run-time verification [31,38]. Basing on content of [8,42,43] one comes to the conclusion, that there is a very small number of developments for industrial verification and specific knowledge domains.

During 2020 - 2024 the interest to practical applications of drones has increased dramatically, and impressive publications have appeared on delivery systems, routing, navigation and verification of real autonomous devices. For example, a method was proposed in for verifying the operation of a drone during task execution, using the knowledge base and rules. An interesting aspect of data protection and routing in delivery drones was touched upon in referring the use of hybrid coding with blockchain components. In a system was proposed for the capturing of illegal drones using a vision and trajectory prediction system. The article discussed the requirement to apply multi-level architecture in the heterogeneous system of agents.

Also agents need combinations of automatic tests and quick visual checks done by human experts. However, neural networks (NN) and deep learning algorithms [49], which already has become the priority tool for computer vision and speech systems [13–15], didn`t propose yet new tools for the imitation of human abilities in verification tasks. Moreover, the known problem of NN to make selective data corrections limits the possibility to use NN in verification procedures. Now computer vision systems based on NN [13,14,49] can slowly enough recognize some components of real scenes of action, and can`t detect illegal physical activity of cheaters and rival robots.

As quantum keys distribution (QKD) schemes were realized not only in fiber optics lines, but also in wireless atmospheric data ones [50], they principally can be used for secured robotic communications at comparatively low keys generation rates and short enough distances. Unfortunately, quantum cryptography is too expensive for low data traffic sessions for data verification in MAS.

In order to design multi-parametric verification models, demonstrating scalable description and navigation in the large-scale structure of parameters of network robotics, one can use multiple–valued logic (MVL) modelling based on discrete Allen-Givone algebra (AGA) taken with a large number of truth levels ($k>128$) [52–54]. Earlier there were attempts to use the version of MVL with several truth levels and to realize Kripke structures [55–57] for the model checking and modeling of time processes, however that has produced too complicated for practice models. The intricacy here is that representitative graph analysis for all possible transitions of system states is problematic to be used in AI systems with the large number of parameters, noise factors, probabilistic and approximate patterns for the description of people, technical objects, and their faults. But this problem seems to be partially bypassed by the integration of traditional precise, probabilistic, and approximate calculations being held within MVL, Boolean, and fuzzy logic. That determines the choice of the heterogeneous logic architecture of agent [52], which is the tool to simplify joint use of various logic models in multi-parametric modelling. Now AGA can held template matching for pattern recognition tasks and may describe neural network expressions [22]. Fuzzy logic formalisms can be emulated by AGA, which was shown to model the analog of the fuzzy controller with parametric T-gates (or T-norms and T*-conorms) [59]. Basing on quantum random number generators, flexible AGA schemes can model one-time cipher pad secret coding scheme [18], random oracle scheme [60], combined classic and quantum protocols for position-based cryptography [53], and also they gave a pair of algorithms for distant comparison of parameters without their disclosing [59]. AGA verification model provides components of blockchain and the distributed ledger, aggregated in collaborating network nodes [53,54]. All these designs creates new possibilities for the distant data verification in collaborating agents.

1.2. Verification and AI Algorithms in Agents

Distant verification of autonomous agent imposes selective monitoring of the complicated system, imitating dozens of basic human abilities and providing self-sustaining of autonomous functioning, activity and reactivity to external world, ability to communicate, the presence of own goals, beliefs and etc. That is why verification of agent is somewhat a wider task, than the model checking in computing [55–57], and it should use many heuristic and semi-empirical methods where the human experts are the final instance. Typical procedures suppose the approval of agent`s authenticity, checks of data integrity, space coordinates confirmation in position-based cryptography, and the verification of passed routes and received licenses [22]. Besides initial debugging, they should include periodic tests, checks after learning and self-learning sessions, upgrade, and data restoration [6–8]. One is preferably to include human biometrics of administrators, and such digital “fingerprints” of robots as parts numbers, random paroles, and the prehistory of earlier done work [22]. However, papers reviewed in sec.1.1 didn`t took enough into consideration AI specifics of agents.

Principally, all mentioned above verification methods are to be integrated into some robotic world model (WM) [62], which can be considered as the set of formal expressions currently describing the interaction of robot with objects, selected in the scene of action by the decision–making system. In the presented paper we don`t discuss in details such a global item as the design of WM [62], but suppose to use for its control the heterogeneous logic model of agent [52], combining AGA with traditional Boolean, and fuzzy logic. One of necessary steps here is the design of the joint ordered structure of robotic terminology and the simple communication logic language of agents, which can describe the logic model of the scene of action for different agents and in prospect to involve natural language processing systems [63].

1.3. Network Verification and the Linked List Scheme

Robotic agents from some allocated MAS are supposed to carry out work in a space scene of action, including various individual partner and rival robots, MASs, people, animals, buildings, vehicles, and other objects. All of them are supposed to be recognized by a computer vision system [13,14], LIDAR [64,65], and other sensors. Here for simplicity we don`t consider dashcam videos, which is a useful tool for verification, but itself needs massive AI-based image procession. In order to simplify the modelling of network verification procedures, one should consider several most typical situations, which can occur for a distant robot and are shown in **Figure 2**. The agent in the allocated MAS can interact with its distant administrator (bot or a man), which is to support distant work of the agent and to verify some of its parameters in a planned mode or for the urgent correction of robot`s behavior. The agent is supposed to exchange any possible communication messages with the administrator (marked by red arrows) and more limited data sets with other agents of the same MAS (shown by yellow arrows), where messages can mainly include parameters of the scene and tasks. The task of data procession here is to provide the logic control for content access, at the same time excluding the transfer of illegal code or confidential information via data channels.

Also, the scene of action imposes the presence of corporative or state regulators with their mobile or static agents, checking licenses, payments, and technical parameters of robots. Agent of MAS should typically exchange with them by several types of formal messages (marked blue arrows) with the limited volume of data.

Verification method should also take into account possible data exchange between MAS and loyal external agents or network nodes, involved by mutual services or a rent. The data flow between agents and external ones (marked by green arrows in **Figure 2**) can contain requests to send some necessary data, tools, or to write/read notations into the copies of the linked list (LL) or the distributed ledger [66–68], which is the distributed protected data storage. Potentially, people in the scene also can communicate with robots concerning positioning, parking and other common problems, which also need special tools for the translation of robotic messages to people and vice versa. Thus, even minimalistic consideration of the scene of action demonstrates the necessity to describe the branched structure of communicating participants and their language protocol.

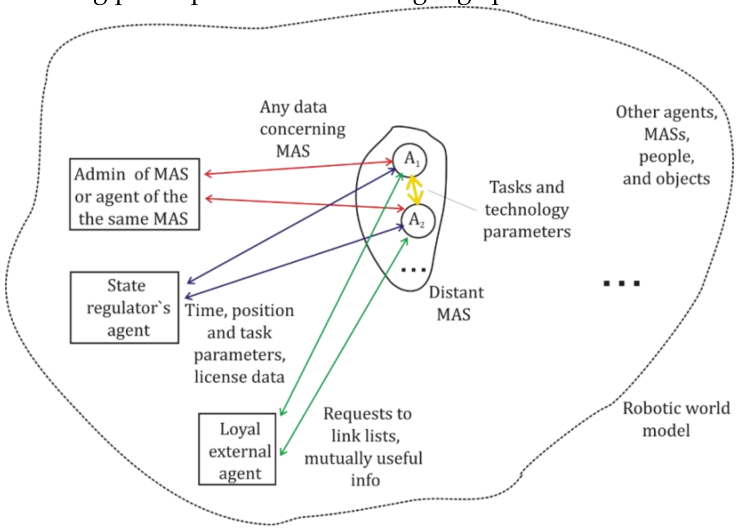

Any data concerning MAS
Admin of MAS or agent of the same MAS
A_1
A_2
Tasks and technology parameters
Other agents, MASs, people, and objects
State regulator`s agent
Time, position and task parameters, license data
Distant MAS
Loyal external agent
Requests to link lists, mutually useful info
Robotic world model

Figure 2. Distant MAS and its agents $A_1, A_2, \ldots$ are considered within some scene of actions, containing other robots and MASs, people, animals and objects.

Also for the efficient interaction with other robots and people an agent is supposed to support and to approve its own trust level (i.e. reputation coefficient) with the help of network "witnesses", involved into the verification of credentials and authorities. Besides collective resources, possible variant here is to use network low-capacity data storages written in collaborating agents and loyal network nodes, and the further discussion is concentrated on the design of coded vocabularies structure, containing necessary basic terminology for dialog communications and for easy data documenting and extraction by means of AGA-based version of the LL [53,54].

In traditional blockchain scheme [66–68] LL prevents illegal data modifications in a group of highly motivated users by sustaining the distributed set of copies in participating network nodes, which are protected by standard tools and hardly can be quickly modified unnoticed by others. In contrast to blockchain, the AGA-based version of LL [53,54] is intended for the collectives of robots and loyal network nodes possessing limited free memory resources. The task here is to "hide" small volumes of critical data, which can later help to detect data errors and modifications. Such resources in agents are considered as hardly accessible ones, as data procession in them is typically optimized for basic operations and only limited data volumes can be periodically extracted by allowed polling procedures.

The formal logic structure of AGA-based LL can be given [53,54] by the function

$$h^{(m,s)} = F_{ll}(m, t, \boldsymbol{e}_m, \boldsymbol{e}_{m-s}, \boldsymbol{h}_m, \boldsymbol{h}_{m-s}) \quad (1)$$

where the input variables are the number m and the time stamp t, vectors $\boldsymbol{e}_m$, $\boldsymbol{e}_{m-s}$ describe the last and the previous entries, and vectors $\boldsymbol{h}_m, \boldsymbol{h}_{m-s}$ are the approving hash values assigned by verifying network nodes. LL has the logic expression structure $h^{(out)}$=$p.t._1 + \cdots + p.t._m$, where e.g., the last written entry can be represented as

$$\begin{gathered} p.t._m = h_m^{(m,1)} \star X_m(m,m) \star X_t(t_m,t_m) \star \\ \star X_{e,1,m}(e_{1,m},e_{1,m}) \star \ldots \star X_{e,1,p}(e_{p,m},e_{p,m}) \star \\ \star X_{h,1,m}(h_{1,m},h_{1,m}) \star \ldots \star X_{h,Q,m}(h_{Q,m},h_{Q,m}) \star \\ \star X_{e,1,m-1}(e_{1,m-1},e_{1,m-1}) \star \ldots \star X_{e,p,m-1}(e_{p,m-1},e_{p,m-1}) \star \\ \star X_{h,1,p}(e_{1,m-1},e_{1,m-1}) \star \ldots \star X_{h,Q,m-1}(e_{Q,m-1},e_{Q,m-1}). \end{gathered} \tag{2}$$

The simplified scheme of AGA-version LL is shown in **Figure 3**, where the pre-agreed collective of mutually loyal robots and network nodes is written in the list of participants and creates equal copies of the collectively formed ledger [53,54]. The formal expression structure for AGA-based LL is shown below in **Figure 3.** Here the first of entries is mixed with the second, and further the second is to be mixed with the third one etc. Thus, LL "mixes" by logic operations random hash values, assigned by external participants, with previous and forthcoming entry vectors $\boldsymbol{e}$. Respectively, MVL version of LL needs some store of quasi random one-time keys [18], preliminary generated by a high quality quantum random number generator QRNG [69]. Like in the blockchain method [66–68], data written in the entry can be approved collectively by the majority voting of participants, whose result is considered as the most reliable one. Certainly, the reliability of such data depends on the number of really involved external participants, and for the small one the task is to introduce additional logic mixing of data taken from various internal sources. Respectively, the advanced version of AGA-based LL can additionally mix parameters of internal subsystems of the agent with external data and assigned hash values.

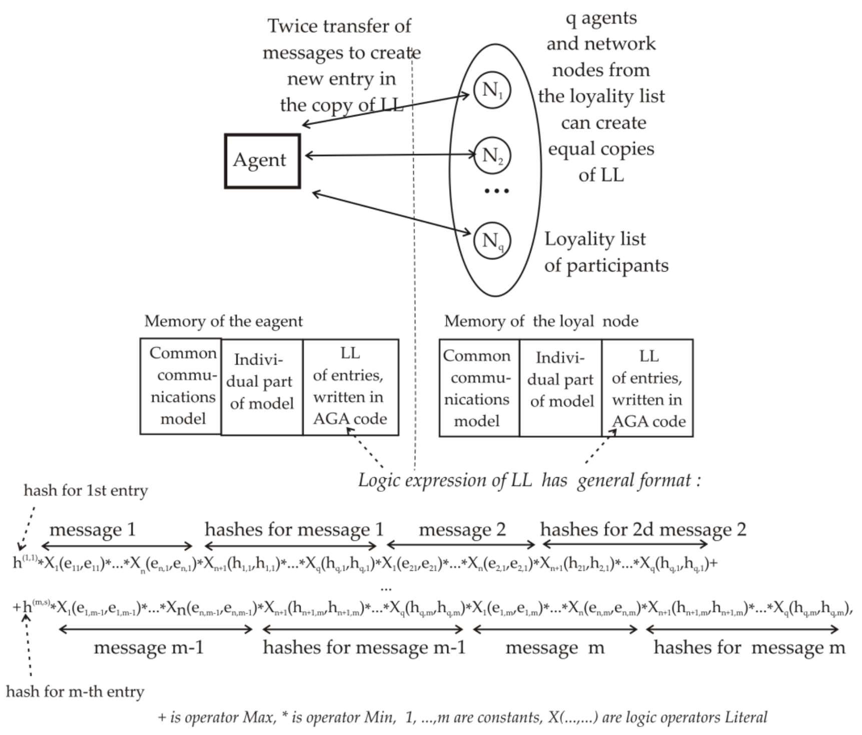

Twice transfer of messages to create new entry in the copy of LL
q agents and network nodes from the loyality list can create equal copies of LL
Agent
N1
N2
Nq
Loyality list of participants
Memory of the eagent
Common commu-nications model
Indivi-dual part of model
LL of entries, written in AGA code
Memory of the loyal node
Common commu-nications model
Indivi-dual part of model
LL of entries, written in AGA code
Logic expression of LL has general format :
hash for 1st entry
message 1
hashes for message 1
message 2
hashes for 2d message 2
h(1,1)*X1(e11,e11)*...*Xn(en,1,en,1)*Xn+1(h1,1,h1,1)*...*Xq(hq,1,hq,1)*X1(e21,e21)*...*Xn(e2,1,e2,1)*Xn+1(h21,h2,1)*...*Xq(hq,1,hq,1)+
...
+h(m,s)*X1(e1,m-1,e1,m-1)*...*Xn(en,m-1,en,m-1)*Xn+1(hn+1,m,hn+1,m)*...*Xq(hq,m,hq,m)*X1(e1,m,e1,m)*...*Xn(en,m,en,m)*Xn+1(hn+1,m,hn+1,m)*...*Xq(hq,m,hq,m),
message m-1
hashes for message m-1
message m
hashes for message m
hash for m-th entry
+ is operator Max, * is operator Min, 1, ...,m are constants, X(...,...) are logic operators Literal

Figure 3. The simplified scheme of AGA-based LL formation, which includes the twice transfer of messages between the agent and network participants of the protocol. The list of possible participants should be formed beforehand and includes agents of the same MAS and partner network nodes.

In order to use AGA–based LL on practice, it is necessary to overcome several obstacles. The first of them is to create the control scheme to support the necessary structure of robotic requests and replies (considered together they are called entries), taking into account the structure of internal modules in various agents. Another challenge is to combine automatic and human checks [63]. One more unsolved problem is to prove that no messages were lost, illegally deleted, or added by cheaters. The last opportunity somewhat intersects with the detection of attacks "man-in-the middle" [70], but in contrast to cryptography now the aim is to prevent data modifications and to imitate human ability to analyze content in the intermittent dialog.

Both version of AGA-based LL [53,54] are oriented at such practical tasks as:

- approval of requested numbers of licenses and permissions,
- following to restrictions declared for cargo parameters and route passing,
- monitoring of visits only to legal zones of the geographical map,
- loading/uploading in legal terminals,
- compliance with the standards of ecology and transport.

1.4. The Aim of the Paper

The goal of this paper is to:

- widen the spectrum of AGA-based data verification algorithms for agents by the holistic vocabulary structure of terminology and its logic coded variables;
- simplify and unify verification schemes by AGA-based communication language, convenient for dialogs of agents in extendable large-scale robotic system;
- adapt the LL scheme to local data storages in network agents and to messages integrity checks;
- provide the clear control of verification schemes in the heterogeneous logic architecture of agent.

Further sec. 2 contains MVL basic mathematical tools, sec.3 proposes AGA representation of the vocabularies structure and the communication language, sec. 4 is devoted to adapted verification schemes based on LL, and sec. 5 presents the fragmentation scheme of AGA functions for human checks. Sec. 6 discusses microassembler modelling and supporting platform. Sec. 7 discusses further research.

2. Method: Basic MVL Model

According to earlier proposed heterogeneous logic model of an agent [52], AGA is considered as a tool to design multi-parametric logic model for the description of the scene of action and verification procedures.

2.1. Multiple-Valued Allen-Givone Algebra

As in earlier publications [52–54], we consider basic definitions of MVL according to [51], where the discrete K-valued Allen-Givone algebra (AGA), operating with discrete MVL functions $y = f(x_1, \dots, x_n)$, is given for n input variables $x_1, \dots, x_n$ and one output variable y. It is very handy for multi-parametrical modelling in AGA, that variables and an arbitrary function $y = f(x_1, \dots, x_n)$ can have K discrete truth levels from the finite set of natural numbers, i.e. $x_1, x_2, \dots, x_n, y \in L = \{0,1, \dots, K-1\}$. Here 0 respond to the absolute false value and $k-1$ indicates the absolutely true one. The complete set of AGA operators given by exp.(1) guarantees the representation of an arbitrary logic function

$$< 0,\ 1, \dots,\ K-1,\ X(a,b),\ *,\ +> \qquad (3)$$

$y = f(x_1, \dots, x_n)$ by some combination of

- logic constants $0, 1, \dots, K-1$,
- operators Min(x_i, x_j) or $\wedge$, marked by $*$, select the minimal truth level in the pair x_i and x_j,
- operators Max(x_i, x_j) or $\vee$, marked by $+$, select the maximal truth level in the pair x_i and x_j,
- operators Literal $X(a,b)$ given by exp. (2):

$$X(a,b) = \begin{cases} 0, & if\ b < x < a \\ K-1, & if\ a \le x \le b \end{cases} \qquad (4)$$

where always $b \geq a$, and $a, b \in L = \{0, 1, \ldots, K-1\}$.

2.2. Possible Discrete Scales for Mapping of Truth Levels

All truth levels in AGA, see **Figure 4** (**a**), are to be processed and compared according to the scale $0, 1, \ldots, K-1$, [51], which belongs to the class of lattices, where all levels are comparable. But principally, MVL may be given differently, e.g., in [56,57] the 6-level logic was defined with two non-comparable elements "don`t know " and "don`t care " for the scale shown in **Figure 4** (**b**). Such model was designed for model checking by means of temporal logic and Kripke diagrams [56], and in fact it included both positive and negative estimations of truth levels. However, in other papers of these authors`s this system has given complicated and non-visual method of checks. In contrast to it, the proposed further method is intended for the heterogeneous logic architecture of agent based on AGA and uses simple scale with the large number K of all comparable truth levels. It can exceed 256 ones and may use the full capacity of standard 8-16-32-64 bitmemory registers for the emulation by microprocessors [72]. Here the axis with truth levels is the scale for mapping of various subsets of physical and model parameters, and we interpret the truth level as the ordered affiliation of different variables and corresponding mapped objects to the holistic agent`s model, as well as the tool to classify parameters. Involved logic parameters may respond to incomparable physical properties (e.g., red and triangle), which should be described by various logic variables mapped of the truth scale given in **Figure 4** (**b**). Principally for wide-band space systems or long-term time control one can use nonlinear mapping onto the scale of truth levels, thus complementing the method of correlated variables discussed in sec. 2.4.

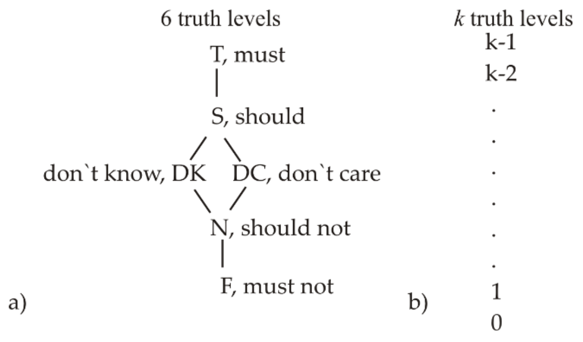


Figure 4. (**a**) Complicated version of MVL proposed in contains 6 truth levels with two non-comparable elements in lattice and was proposed for the design of temporal logic and Kripke structures. (**b**) Simple version of the linear truth levels scale was used in AGA [51], which is more convenient for the description of large-scale models of intellectual agents.

2.3. Formation of AGA Functions

For holistic or fragmented MVL modelling one should use traditional and universal representation of MVL function by the truth table shown in Table 1. MVL function $y = f(x_1, \ldots, x_n)$ has the overall number of raws $K^n - 1$ in contrast to Boolean logic with its only $2^n - 1$ rows. The column for the output variable $f(x_1, \ldots, x_n)$ should be filled in by logic constants $C = \{0, 1, \ldots, K-1\}$ according to expert choice or some formal model. Every row of such truth table with the nonzero value of the output variable $F(\ldots, \ldots, \ldots)$ has equivalent representation by a product term [51], written via logic constants and operators from exp. (5).

Table 1. Basic format of the truth table defined in AGA [51].

N_{row}	Input		variables				Output variable
	x_1	x_2	x_3	x_4	…	x_n	$F(x_1, \ldots, x_n)$
0	0	0	0	0	…	0	$F(0, \ldots, 0)$
1	1	0	0	0	…	0	$F(1, \ldots, 0)$

…	…	…	…	…	…	…	…
K^n-1	K-1	K-1	K-1	K-1	…	K-1	$F(K-1,\dots,K-1)$

The overall logic expression is given by exp. (5):

$$y = F(0,0,\dots,0) * X_1(0,0) * X_2(0,0) *\dots* X_n(0,0) + \\ +F(0,0,\dots,1) * X_1(1,1) * X_2(0,0) *\dots* X_n(0,0) + \quad (5) \\ \dots \\ +F(K-1,K-1,\dots,K-1) * X_1(K-1,K-1) * X_2(K-1,K-1) *\dots* X_n(K-1,K-1).$$

Here indexed Literals $X(\dots,\dots)$ are written by $X_j(q,q)$ and filled in by equal parameters $a=b=q$, where q values are taken from the corresponding row of the truth table. For memory storage and calculations, AGA functions can be equivalently represented by matrixes exp. (6), formed by logic constants $C=\{0,1,\dots,K-1\}$ and describing values of the output variable $F(\dots,\dots,\dots)$ and indexed pairs of Literal parameters (a_i,b_i), $i=1,\dots,n$:

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots \\ a_{K^n-1,1} & \cdots & a_{K^n-1,n} \end{pmatrix},\ B=\begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \cdots & \cdots & \cdots \\ b_{K^n-1,1} & \cdots & b_{K^n-1,n} \end{pmatrix}, C = \begin{pmatrix} c_{11} & \cdots & c_{1v} \\ \cdots & \cdots & \cdots \\ c_{K^n-1,1} & \cdots & c_{K^n-1,v} \end{pmatrix}. \quad (6)$$

Here always $\boldsymbol{b_{ij}} \geq \boldsymbol{a_{ij}}$, n - is the number of input variables, $\boldsymbol{K}$ – is the number of truth levels. The switching function in AGA guarantees the reproducing only of data written in its truth table or in equivalent logic expressions, and can output error only if it was illegally or not correctly modified.

2.4. Correlated Variables for the Description of Large-Scale Space and Time Bands

Mobile agent should imitate human ability to estimate time, space, light intensity, and some other parameters in wide bands of values and with various precision (e.g., from micrometers to thousands of kilometers). In order to obtain wide range of measurements, to avoid too large number of truth levels K, and to use more cheap hardware, one can use the method to combine in one AGA logic expression data, describing the same physical parameter with various precision of data in the same function. Then the description of e.g., position x =2830 m for the space axe $[0,x)$ can be represented in the digital map as $x = 2\ km + 800\ m\ + 30m$ and written by an expression like $x = x^{(1)} \times 1 + x^{(2)} \times 10 + \cdots x^{(p)} \times 10^p$. This expression can be represented in the MVL truth table with any necessary precision as a set of p correlated input variables $x^{(1)}, x^{(2)},\dots,x^{(p)}$. Such variables should be processed as ordinary logic variables in the multi-parametric AGA function [52], but only the summation of the whole set of used correlated parameters provides correct restoration of physical distances. This step should be monitored in methods described in sec. 3 and 4 for forward and backward mapping between truth levels representation and natural numbers one.

For description of various systems, one can freely exploit correlated variables together with ordinary ones, and there is no necessity to insert all defined correlated variables $\boldsymbol{t}^{(k)} = \{t^{(1)},\dots,t^{(K)}\}$ into all used functions, as namely the shortened set of correlated variables is convenient for human or quick automatic checks. If necessary, e.g., for monitoring of 2D space maps, one can use input space variables $\boldsymbol{x}^{(i)} = (x^{(1)},\dots,x^{(I-1)})$, $\boldsymbol{y}^{(j)} = (y^{(1)},\dots,y^{(J-1)})$, and time variables $\boldsymbol{t}^{(k)} = (t^{(1)},\dots,t^{(K-1)})$ in a separately defined MVL function $F\left(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(j)},t^{(3)}\right)$, which may describe time processes ranging from nanoseconds in microcontrollers up to months in a logistic navigation system for the description of visits to cargo terminals.

However, the use of correlated variables has some restriction disclosed by the **Figure 5** for the correlated space variable $x^{(l)}$. The excessive number of chosen correlated input variables $x^{(l)}$ will finally lead to the situation, when the resulting measurements and the statistical error will achieve the given difference between two neighbor truth levels. Then according to the theory of physics measurements [72], both truth levels are possible within the given confidence interval, and AGA model then should add two new product terms $C * X_x^{(l)}(a,b) * \dots$ with logic constants C equal to 1 and 2. This contradicts to the principal of AGA modelling of control systems, which needs bijective mapping of physical and logic values for forward and reverse interpretation of data. Such ambiguity can lead to errors and needs to shorten the excessive number of correlated variables in case (**b**); this restriction should be taken into account at least for space coordinates, time, and laser beam intensity. AGA calculus hasn`t embedded tools to check such data, and the designer should do this himself.

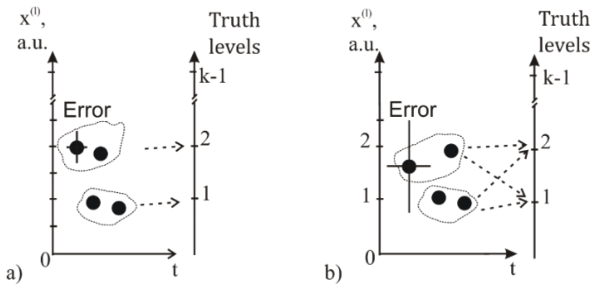


Figure 5. Data error limitation for the choice of correlated variable $x^{(l)}$. (a) Correct mapping onto the scale of truth levels. (b) Case (**b**) differs from (**a**) by the large error value, where each of two groups of data can be classified both as 1 or 2. Incorrect mapping occurs, if the statistical and measurement error δ exceeds the difference between two neigbour truth levels, i.e. $\delta \leq k_j - k_{j-1,}\, j = \{2, \dots, K-1\}$.

3. Results: AGA Logic Variables for Scene Description and Dialog Communications

Further the method is proposed to describe robotics scene of action and dialogs of agents within the discrete logic calculus AGA, using the large number of discrete truth levels k ≥ 256 [18,52–54] and the ordered structure of coded vocabularies containing selected basic terminology. Partially it dates back to ontology methods reviewed in [8].

3.1. Representation of the Scene of Action by Natural Language, Numerical Codes, and Truth Levels

According to sec. 1.3 we consider the agent and its MAS within some scene of action (see **Figure 2)** containing various partner and rival participants. Necessary definitions are given in **Table 2.** It considers the set (or the total vocabulary) V^L of all different robotic terms, given by words or their collocations selected by knowledge experts. These vocabularies should describe all input and output variables used in model functions.

Table 2. Basic formal parameters for the description of the scene of action.

Notation	Definition
K	Finite set of truth levels K is defined in AGA [x], where $k = \{0,1,2,\dots,K-1\}$, $K \subset N$, N – is the set of natural numbers. Only finite sets are intended for basic logic AGA modelling and the mapping of words vocabularies.
V^N	Finite subset v^N of the set of natural numbers, $v^N = \{1,2,\dots.,K\}$, $V^N \subset N$. It is intended for auxilliary representation and mapping of words vocabularies.
V^L	The finite set (or the vocabulary) of all selected different robotic terms, given by words and word collocations of natural language. It describes the scene of action, robotic tasks, mathematical and verification procedures.
V_v^K	Modelling of the scene of action uses v subsets of truth levels, $v = \{1,\dots,K\}$, obtained by arbitrarily chosen bijective mapping $V^N \rightarrow K$, i.e. onto the scale of truth levels $k = \{0,1,\dots,K\}$ defined in AGA.
V_v^N	Subsets V_v^N of vocabulary V^N are obtained by arbitrarily chosen mapping of $V^L \rightarrow V^N$, i.e. onto the scale of natural numbers $n \in N = \{1,2,\dots,K\}$, where the set V^N consists of v subsets $V_v^N \in V^N, v = \{1,\dots,K\}$;

V_v^L	Subset V_v^L of the vocabulary V^L determines one of v given classes of robotic terms, containing only different elements, $V_v^L \subset V^L, v = 1, \dots, V, V \leq K$, where K - the maximal number of truth levels. In majority of tasks convenient classes of robotic terms correspond to classes of words (noun, verb, adverbial modifiers of place, time etc.)
w_v^L	Element of a subset (or vocabulary) V_v^L can be given by a word or word collocation, representing in natural language some robotic term.
w_v^N	Mapping of w_v^L onto the subset of natural numbers V^N, providing number code representation of a robotic term.
w_v^K	Mapping of w_v^L onto the set K, providing equivalent logic representation of a robotic term.
$\boldsymbol{W}_z$	Word phrase is a vector $\boldsymbol{W}_z$=(w_1, w_2, …, w_Z) with declared fixed position of elements according to their numbers v={1,..,Z}, $Z \leq V \leq K$. Word phrase is a sampling of only one word from the next vocabulary. For various types of procession it can be written by elements w_v^L, w_v^N, w_v^K of subsets V_v^L, V_v^N, V_v^K. Note that the list of selected vocabularies and message format can differ for various classes of tasks.

One should note, that the code word may mean action, tool, some equation, and subroutine for calculations of robotic work parameters. Total vocabulary can be preliminary selected by library services and processed by experts [73]; here we omit possible problems for the translation from one language to another one. The selected collection of necessary terms V^L is to be subdivided into subsets V_v^L, describing robotic tasks, actions, and tools by various word classes (noun, verb, adjective, cardinal numbers etc.). Then we should represent all these terms by unique natural numbers codes written in vocabularies V_v^N. This intermediate coding is necessary for convenient grouping of terms and verification procedures. Further one should map these number codes onto the discrete scale of truth levels K defined in AGA. Corresponding structure of mapping is shown in **Figure 6**.

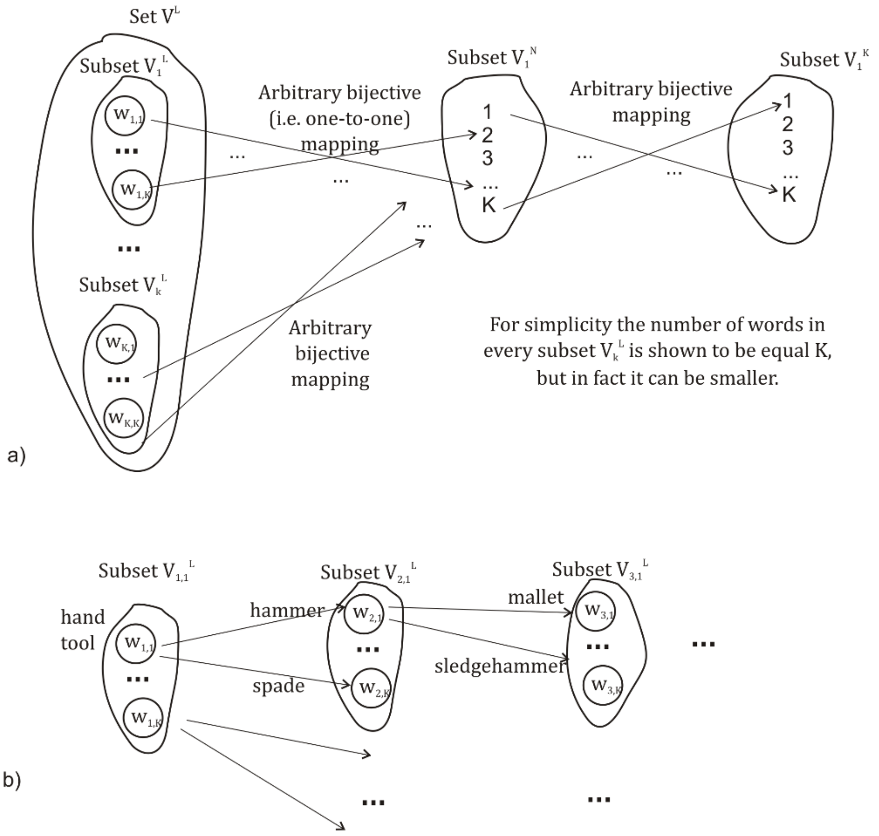

Set V^L
Subset V_1^L
w_1,1
w_1,K
Subset V_k^L
w_K,1
w_K,K
Arbitrary bijective (i.e. one-to-one) mapping
Subset V_1^N
1 2 3 … K
Arbitrary bijective mapping
Subset V_1^K
1 2 3 … K
Arbitrary bijective mapping
For simplicity the number of words in every subset V_k^L is shown to be equal K, but in fact it can be smaller.
a)
Subset V_1,1^L
hand tool
w_1,1
w_1,K
Subset V_2,1^L
hammer
spade
w_2,1
w_2,K
Subset V_3,1^L
mallet
sledgehammer
w_3,1
w_3,K
b)

Figure 6. Description of the scene of action by root and sequential chain vocabularies. (**a**) Mapping of arbitrarily selected root vocabularies $V_{1,1}^L \rightarrow V_{1,1}^N \rightarrow V_{1,1}^K$, transferring natural language terminology

onto scales of natural numbers and logic truth levels; **(b)** Chain mapping of vocabularies like $V_{1,1}^{L} \rightarrow V_{2,1}^{L} \rightarrow V_{3,1}^{L} \ldots$ is convenient for the content clarification and formation of structured word phrases. Respectively, every vocabulary $V_{v,m}^{L}$ further has double index for compact matrix notation.

The specifics of mapping to V_v^K in **Figure 6 (a)** is that we can arbitrarily or randomly assign truth levels to words in order to obtain convenient for the classification groups of terms [52], as the simplification (and shortening) of AGA expressions is based on consensus method of minimization [51], which ranks and forms logic product terms according to logic constants. Substantial feature of the used subsets V_v^L, V_v^N, V_v^K is that the number of elements in them is less or equal K, and unused vacancies in subsets are to receive zero values. Mappings $V^L \rightarrow V^N$ and $V^N \rightarrow V^K$ are bijective, i.e. each element of a subset maps to only one of elements in another subset, what provides simple forward and backward translation by transcoding tables. Chain mapping scheme is disclosed by **Figure 6** (**b**) and responds to sequential clarification of content by the finite chain of conjugated vocabularies $V_{1,m}^{L} \rightarrow V_{2,m}^{L} \rightarrow V_{3,m}^{L} \ldots \rightarrow V_{K,m}^{L}$, where $V_{1,m}^{L}$ is a root vocabulary. Such method is to provide the unified protocol both for robots with simplest and advanced hardware platforms, where the first part of the message can contain coded words written closer to root vocabularies and the actuality of information can be quickly estimated by all agents, even capable only to partially interpret the message. Also agents can undertake some actions immediately, before the transfer of all details.

The formation of content root vocabulary V^L is given in **Table 3,** which describes words of natural language. Every term (or word combination) of natural language is included into some vocabulary. Every vocabulary $V_{v,m}^{L}$ contains up to K different words $w_{v,m}^{L}$. Numbers $N = 1 \div M$ refer to root vocabularies. Values N from $K+1$ up to $2K$ refer to the second one in the chain mapping of vocabularies.

Table 3. The structure of vocabularies $V_{v,m}^{L}$ initially given in natural language is described as a matrix KxM, where K is the initially defined maximal number of truth levels in AGA.

N	Name of Vocabulary	Content	Class of Word In Natural Language	Natural Language Representation (Maximal number of words - *K*)
1	$V_{1,1}^{L}$	Initiation of dialog and format of message	Cardinal Number	{one, two, …, *K*}
2	$V_{1,2}^{L}$	Fixed parole	Cardinal Number	{one, two, …, *K*}
3	$V_{1,3}^{L}$	Hash value	Cardinal Number	{one, two, …, *K*}
4	$V_{1,4}^{L}$	Addresser of message	Noun	{administrator, robot of MAS, external robot, man, vehicle, unidentified object}
5	$V_{1,5}^{L}$	Sender of message	Noun	{administrator, robot of MAS, external robot, man, vehicle, unidentified object}
6	$V_{1,6}^{L}$	Relative time of sending	Cardinal Number	{one, two, …, *K*}
7	$V_{1,7}^{L}$	Time of action	Adverbial modifier of time	{Immediately, now, near future, in an hour, at the specified time…}
8	$V_{1,8}^{L}$	Action/ Task	Verb	{verify, measure, read, write, move to, upload, download, plugs/connectors check, software check, circuit board test, coating check, mechanics check, …}
9	$V_{1,9}^{L}$	Object of action	Noun	{admin, addresser, robot of MAS, external robot, man, house, technical construction, vehicle, road, tree, animal, pit, vocabulary, unidentified object,…}
10	$V_{1,10}^{L}$	Number of object of action	Cardinal Number	{one, two, …, *K*}
11	$V_{1,11}^{L}$	Place of action/ objects of action	Noun, Adverbial modifier of place	{ robot, external object, vehicle, internal module, …}
12	$V_{1,12}^{L}$	Reference object for relative coordinates	Noun	{robot of MAS, external robot, man, house, technical construction, vehicle, road, tree, bush, animal, pit, stone, …}
13	$V_{1,13}^{L}$	Place of action/ coordinate x	Cardinal Number	{ GPS coordinate x, relative coordinate one, two, …, *K* }

14	$V^L_{1,14}$	Place of action/ coordinate y	Cardinal Number	{one, two, …, K }
15	$V^L_{1,15}$	Linked List	Cardinal Number	{one, two, …, K}
16	$V^L_{1,16}$	Natural number code	Cardinal Number	{one, two, …, K}
17	$V^L_{1,17}$	Truth level code	Cardinal Number	{one, two, …, K}
18	…	…	…	…
M	$V^L_{1,M}$	…	…	…
…	$V^L_{2,1}$	Format of message	Cardinal Number	{one, two, …, K}
…	$V^L_{2,2}$	Fixed parole	Cardinal Number	{one, two, …, K}
…	$V^L_{2,3}$	Addresser of message	Noun	{administrator, robot of MAS, external robot, man, vehicle, unidentified object}
…	…	…	…	…
2M	$V^L_{2,M}$	…	…	…
…	…	…	…	…
KxM	$V^L_{K,M}$	…	…	…

The first field in **Table 3** is the initiation (or welcome) word of dialog, also indicating the type of communication format for messages and the number of parts of the phrase, which may differ for various sectors of robotics. The name description of addressers and senders may use matrix $V^K_{v,m}$ with chain vocabularies and several bytes for microprocessor representation of participants, but for brevity only two chain vocabularies $V^K_{v,2}$ are shown in the **Table 3**; their common matrix is defined by two parameters $v, m \le K$. For simplicity, further we avoid (where possible) to use the third index for the description of the number of a word in the specific vocabulary. The order of numbering in vocabularies $V^L_{i,j}$ is not substantial. Every tool and identified object in the scene of action can have only one name. Note e.g., that the time of action w^K_7 (from $V^K_{1,7}$ in **Table 3**) can be taken with symbolic values "immediately" or "urgent".

AGA modeling of the scene of action is based on three equivalent representations of vocabularies $V^L_{v,m} \rightarrow V^N_{v,m} \rightarrow V^K_{v,m}$ shown in **Table 4.** The parallel use of natural language, natural numbers code and truth levels naturally occurs when one firstly tries to explain the work of the designed robotic system, and after this turns to numbers codes and AGA models. Such rigid binding of variables representations is convenient for human experts.

Table 4. The equivalence of words in vocabularies $V^L_{v,m}$, $V^N_{v,m}$, and $V^K_{v,m}$.

N	**Natural language Vocabulary**	**Natural numbers code of Vocabulary**	**Truth levels code of Vocabulary**	**Content**
1	$V^L_{1,1}$	$V^N_{1,1}$	$V^K_{1,1}$	Format of message
	$w^L_{1,1,1}$	$w^N_{1,1,1}$	$w^K_{1,1,1}$	"SOS"
	$w^L_{1,1,2}$	$w^N_{1,1,2}$	$w^K_{1,1,2}$	"Hi, read 2parts message"; *Initiation of the new phrase*
	$w^L_{1,1,3}$	$w^N_{1,1,3}$	$w^K_{1,1,3}$	"Hi, read 3parts message";
	…	…	…	…
	$w^L_{1,1,7}$	$w^N_{1,1,7}$	$w^K_{1,1,7}$	" Ready to continue"
…	…	…	…	…
2	$V^L_{1,2}$	$V^N_{1,2}$	$V^K_{1,2}$	Fixed parole
…	…	…	…	…
KxM	$V^L_{K,M}$	$V^N_{K,M}$	$V^K_{K,M}$	Content given by expert

Correlated variables can be written to **Table 4** as a sequence of $w^K_{i,g,k}$, sequentially raising the precision of space coordinates but following the restriction from sec. 2.4.

The scheme of logic transformations between natural numbers code and truth levels ones is given in **Figure 7**, which shows the initial mapping of $w^N_{i,j}$ data onto the logic representation $w^K_{i,j}$, and further the reverse mapping to natural numbers, as arithmetic check coefficients calculations are to be done within Boolean logic. For further convenient procession, the maximal logic value $K-1$ and its natural numbers equivalent are reserved for possible minimization procedures in AGA. By default, zero value indicates the not used word in a vocabulary. Check coefficients $C_{i,j}^{N}, C_{i,j}^{K}, C_{i,j}^{NXR}, C_{i,j}^{KXR}$ are to be calculated within the mapping of AGA to natural numbers scale (see sec. 3.3).

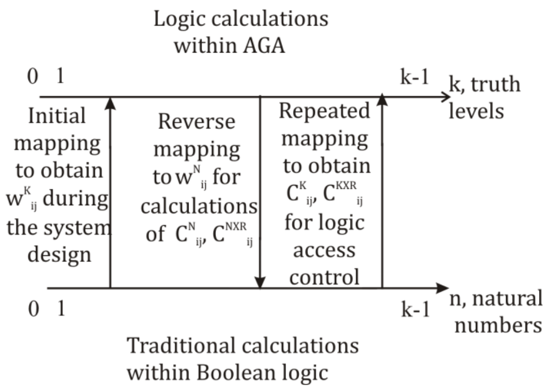
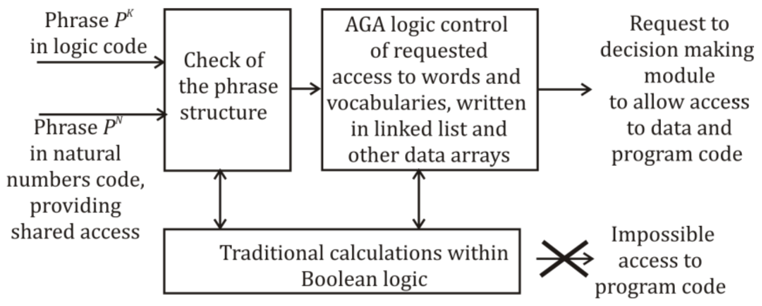

Logic calculations within AGA
0 1
k-1
k, truth levels
Initial mapping to obtain w^{K}_{ij} during the system design
Reverse mapping to w^{N}_{ij} for calculations of C^{N}_{ij}, C^{NXR}_{ij}
Repeated mapping to obtain C^{K}_{ij}, C^{KXR}_{ij} for logic access control
0 1
k-1
n, natural numbers
Traditional calculations within Boolean logic

Figure 7. The scheme of transitions from natural numbers code to logic truth levels code is necessary for the heterogeneous logic architecture of agent. Equal scales for natural numbers and truth levels make this transition just a formal step.

Verification procedures can involve two data input channels with truth level codes and natural numbers ones shown in **Figure 8 (a).** It can include procession and comparison of data coded as in truth levels, as in natural numbers codes. Independent coding of these two channels gives the principal possibility to compare separately final and intermediate results and to detect errors or illegal modifications of data, as final results in both representations should be the same or close, if approximate estimations were used. Also equivalent representations are considered as the binding tool for the heterogeneous logic architecture of agent [52], which supposes parallel calculations in various logic models.

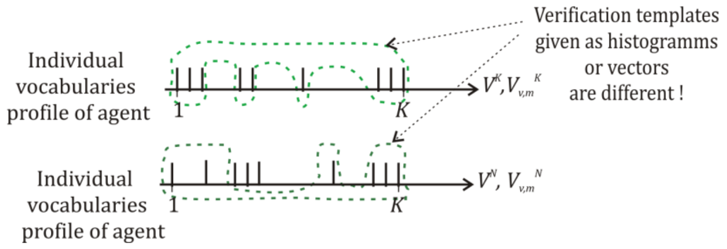
At the same time, natural numbers code and AGA truth levels representations have different numbers profiles for comparison. **Figure 8 (b)** shows the principle of brief verification procedure for used vocabularies in the agent. Besides this, visual representation by histogram shown in **Figure 8 (b)** is more convenient for human checks. Within the proposed scheme of vocabularies, known types of activity and coded tasks execution should lead to preliminary known set of vocabularies describing controllers and actuators. Independent checks may be held both in vocabularies $V_{v,m}^{N}$ and $V_{v,m}^{L}$. As AGA truth levels representation $w_{v,m}^{K}$ is compatible with the secret data coding scheme and the scheme for distant comparison of data without their disclosing [53,54], namely it is preferably to be used for the control of data access in agents. Natural numbers codes $w_{v,m}^{N}$ are better to be used for common access content and less confidential data. Full check of the agent will need comparison of all words $w_{v,m}^{K}$ and $w_{v,m}^{N}$ in used matrixes $V_{v,m}^{K}$ and $V_{v,m}^{N}$. However, all such procedures need one more critical component.

a)



Phrase P^K in logic code
Phrase P^N in natural numbers code, providing shared access
Check of the phrase structure
AGA logic control of requested access to words and vocabularies, written in linked list and other data arrays
Request to decision making module to allow access to data and program code
Traditional calculations within Boolean logic
Impossible access to program code

b)



Verification templates given as histogramms or vectors are different !
Individual vocabularies profile of agent
1
K
$V^K, V_{v,m}^K$
Individual vocabularies profile of agent
1
K
$V^N, V_{v,m}^N$

Figure 8. (a) Two channels to provide separate access to internal data by more protected AGA truth levels codes and by more simple natural numbers codes. **(b)** The principle of brief verification of vector components histogram in possessed and needed profiles for root vocabularies V_v^N , V_v^K and further chain ones.

3.2. Communication Phrase

There exist formal theories of language construction like considered in [74], although modern natural language procession systems typically use Chat GPT and NN [75]. However, earlier the simplest AGA-based scheme of the communication protocol was proposed in for secretly coded instructions transfer via the wireless laser data line $\lambda = 0.63$ µm with pulse-width-modulation. This scheme has used the fixed position of several specialized vocabularies in the phrase $\boldsymbol{W}$ and was intended for 8-bit microcontrollers platforms. The small number of internal memory registers in them has limited the transfer of short phrases given by the format {Sender of message - Receiver of message - What to do - Where to do - When to do - Commentary for next phrase}, coded by 8–bit numbers. This earlier version of protocol corresponds to the shortened sampling taken from **Table 3** for the particular set of vocabularies:

{ V_2^L (Sender of message) $-V_3^L$(Addresser of message) $-V_4^L$(Task) $-V_6^L$(Place of action$-V_{13}^L$(Time of action) $-$ V_1^L(Format of next message)}.

However, now the task is to propose the flexible and unified 8-16-32-… phrase format, adaptable to various tasks and the large number of vocabularies. Also, such format should help to find correct parts of phrase and to determine lost ones which are to be repeated. Additional requirement is to use independent microcontroller data channel for dialogs, providing enough content transfer but using rigidly given protocol, excluding uncontrolled intrusion of external code into the software of agent.

The necessary structure of the phrase vector $\boldsymbol{W^K}$, describing the robotic message, is given in **Table 5.** Every phrase is a fragmented one and contains the addressing header and content parts. Their length is 8-bytes what is accessible for cheap 8-bit platforms, and the overall length of $\boldsymbol{W^K}$ principally can be enlarged up to K parts, but is supposed to be typically limited by 2-8 ones. Components of vector $\boldsymbol{W^K}$ are the elements of various vocabularies given in **Tables 3** and **4**. Principally, alternative representations by words $w_{v,m}^N$ also can be used in phrases, but in this paper we don`t use them. Parts of phrase shown in **Table 5** can be transmitted separately and afterwards may be assembled into holistic phrases with the help of hash values and total check coefficients. That excludes the possible problem of time delays for multi-channel data aggregation.

The header part includes obligatory fields "Format", "Fixed parole", "Addresser", and "Sender" with the fixed position in the phrase for the quick and simple procession. Priority of position of root and other vocabularies in the phrase is given by the field "Format". The value $w^K_{1,1}$ = "Hi" may be given by several various numbers, initiating the new dialog and indicating at the same time the number of content parts. The field "Fixed parole" is intended for the initial access of preliminary allowed agents and includes words taken from the vocabulary $V^K_{1,2}$, which contain fixed passwords for initial contact only. Further communication modules of Sender and Addresser should assign quasi-random hashes to content parts of the phrase. Respectively, every content part also includes obligatory field "Hash", describing assigned one-time quasi-random hashes, necessary for correct integration of transferred parts into the holistic phrase. This field should use preliminary stored reserve of random keys, obtained from external QRNG or random oracle scheme [60]. Such one-time access keys e.g., also were envisaged by AGA protocol for confidential distant comparison of parameters without their disclosing [59]. Fields "Arbitrary code word, $w_{v,m}$" in content parts of the phrase are to be filled in by arbitrarily chosen words, forming some sampling $K^`$ of logic variables, where $K^` \leq K$. The maximal number of content parts j in the phrase is determined by the expression

$$j = N_{used\ vocab.in\ phrase} - N_{used\ in\ header}/N_{free\ fields\ in\ cont.part} = (KxM - 4)/N_{free\ fields\ in\ cont.part} \qquad (7)$$

Format of messages shown in the **Table 5** certainly can be further adapted for a specific knowledge domain, a large number of participants, and for large space and time bands of work**.** E.g., for extraordinary situations like a fire, any robot should have reserved code word in the field "Format" for the activation of chain vocabularies, describing phrases like {Fire service - Fire department q - Fire robot qq-….} and {City q – Municipal district qq - Street qqq-…}. But for routine tasks the long chain of vocabularies can be excessive.

Total check coefficients C^K and C^N shown in **Table 5** need special commentary, as they suppose arithmetic summation of words codes for messages integrity verification. Such verification procedures for a noisy data line can determine, if some parts of the phrase were modified, added, or losted due to noise or intruders. This method suppose summations of codes for all used indexes v, m and parts of phrase. As the summation is not initially defined in AGA and one can`t directly use Boolean calculations for the set of truth levels data, words codes should be preliminary mapped onto the scale of natural numbers N before traditional arithmetic calculations. Results of calculations should coincide for sums obtained by Addresser and Sender. For 8-bit microcontrollers with embedded arithmetic procedures this check unavoidably uses the summation with case overflow, what provides the independence of verification parameter from the number of actually used words and transmitted parts of the phrase. Besides this, in such scheme well-known binary operations XOR used in may be also applied for the detection of permutations of words in a phrase and parts of the phrase. In order to use such verification schemes, one should make several definitions.

Table 5. Word phrase $\boldsymbol{W^K}$ is to be written by the header and several content parts in the logic code, and formed by words taken from some sampling $K^`$ of all defined KxM vocabularies. Notation $w^K_{v,m}$ indicates a word with arbitrarily given indexes v, m, chosen by expert or decision maker from some $V^K_{v,m}$.

1st 8-byte part- header								
Variable	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Name	Initiation of dialog /Format, $w_{1,1}$	Fixed parole,$w_{1,2}$	Addres-ser name 1, $w_{1,4}$	Addres-ser name 2, $w_{2,4}$	Addres-ser name 3, $w_{3,4}$	Sender name 1, $w_{1,5}$	Sender name 2, $w_{2,5}$	Sender name 3, $w_{3,5}$
Vocabulary *(in Table 3)*	$V^K_{1,1}$	$V^K_{1,2}$	$V^K_{1,4}$	$V^K_{2,4}$	$V^K_{3,4}$	$V^K_{1,5}$	$V^K_{2,5}$	$V^K_{3,5}$

				2d 8-byte part- content				
Variable	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
Name	Assigned hash, $w_{1,3}$	Arbitra-ry word, $w_{v,m}$	Arbitra-ry word, $w_{v,m}$	…	…	Arbitra-ry word, $w_{v,m}$	Arbitra-ry word, $w_{v,m}$	Total check coefficient of all previous parts and the current one, C^K or $C^{N/K}$
Vocabulary	$V^K_{1,3}$	$V^K_{v,m}$	$V^K_{v,m}$	…	…	$V^K_{v,m}$	$V^K_{v,m}$	$V^K_{1,17}$
…	…	…	…	…	…	…	…	…
				(KxM-4)/2 th 8-byte part- content				
Variable	x_{K-8}	x_{K-7}	…	…	…	x_{K-2}	x_{K-1}	x_K
Name	Assigned hash, $w_{1,3}$	Arbitra-ry word, $w_{v,m}$	Arbitra-ry word, $w_{v,m}$	…	…	Arbitra-ry word, $w_{v,m}$	Arbitra-ry word, $w_{v,m}$	Total check coefficient of all previous parts and the current one, C^K or $C^{N/K}$
Vocabulary	$V^K_{1,3}$	$V^K_{v,m}$	$V^K_{v,m}$	…	…	$V^K_{v,m}$	$V^K_{v,m}$	$V^K_{1,17}$

Definition 1: Total check coefficient C_i^K of mapped truth levels is given for a phrase $\boldsymbol{W}_i$ written in AGA logic codes and is the arithmetic summation $C_i^K = \sum_v^{K`} w_v^K$, carried out within Boolean logic for truth values of all words $w_v^K$, used in the phrase $\boldsymbol{W}_i$ and given by arbitrary sampling of vocabularies $V^{K`}$, $K`\leq K$. AGA truth levels before summation are mapped onto the linear scale of natural numbers N.

Application of the def. (3.1) is supposed to be convenient both for independent human checks and for automatic comparison with template data.

Another convenient coefficient for checks is given by Definition 2**.**

Definition 2: Total check coefficient $C_i^{N/K}$ of arithmetic summation of natural numbers codes is given for a phrase $\boldsymbol{W}_i$ written in mixed natural numbers and truth levels codes and is the arithmetic summation $C_i^{N/K} = \sum_v^{N`} w_v^{N/K}$, carried out for natural number codes and mappings of truth levels codes of words $w_v^N$ or $w_v^K$, used in the phrase $\boldsymbol{W}_i$ and taken from the sampling set of vocabularies $V^{N`}$ and $V^{K`}$, $N`, K` \subset N$.

Coefficient $C_i^{N/K}$ is necessary for human or automatic checks within the heterogeneous logic architecture of agent, where one can independently visualize natural numbers codes and evaluate the integrity of the dialog data exchange by the summation of all values w_v^N in the phrase.

One more useful tool for the logic verification in agent is the well-known Boolean logic operation XOR, used for the set of words in the phrase $\boldsymbol{W}_i$ represented by truth levels or natural numbers code. Its destination is to make digital "fingerprints" for phrases and their parts. XOR operation can help to compare identity and to hide real codes used in sessions held even without secret coding.

Definition 3: Total coefficient C_i^{XRK} of check XOR binary logic operation taken for mapped truth levels is the set of Boolean XOR operations $C_i^{XRK} = \widehat{w_1^K} \oplus \ldots \oplus \widehat{w_{K`}^K}$ carried out in the phrase $\boldsymbol{W}_i$ for binary representation of AGA truth levels of words $\widehat{w_v^{K`}}$, mapped onto the linear scale of natural numbers N.

Definition 4. Total coefficient C_i^{XRN} of check XOR binary logic operation is the set of Boolean XOR operations $C_i^{XRN} = w_1^N \oplus \ldots \oplus w_{K`}^N$ carried out in the phrase $\boldsymbol{W}_i$ for binary representation of natural numbers codes of words $w_v^{K`}$.

One should note, that coefficients Def. 1÷4 also can be easily used in phrases $\boldsymbol{W}_i$, transmitted by parts, like as it is shown in **Table 4**. "Sewing" or ordering of parts of the phrase $\boldsymbol{W}_i$ in the addresser agent supposes the check of coefficients Def. 1÷ 4 as for whole phrases, as for their separate parts. The only necessary complication here is to insert separately calculated coefficients in the end of every phrase part (see **Table 6**), and every time to fill in the total sum for all parts from the very beginning of the phrase, including the header. The specifics of coefficients Def. 3÷ 4 is that they are intended for more confidential data or for work in the tense media, when the increased level of caution is to be activated in the MAS.

The detection of lost and mixed content parts during the transfer of phrases in dialog mode can be based on the check for assigned one-time hash values and the comparison of data, obtained by the summations of coefficients $C^K_{i.j}$. The example of such algorithm is given in sec. 3.3. Besides this, for vocabularies $V^N_{v,m}$, where $v \in \{1, \dots, K\}, m \in \{1, \dots, M\}, M \leq K$, one can apply coefficients C^N and C^{NXR}, taken in natural numbers representation.

3.3. Communication Module and the Dialog Protocol

Procession of dialog messages is considered further only within the specialized communication module, but for correct design of verification procedures it is necessary to discuss briefly its interaction with other agent`s components. The very simplified interpretation scheme of agent and its basic work cycle is shown in **Figure 9**, although its internal modules can include additional internal feedback contours and auxiliary data lines. Any realistic autonomous mobile robot should include:
a) sensors and computer vision system, which are to detect objects and dangers, providing the agent`s reactivity to the external world and space positioning;
b) module for the procession of messages, coming from radiofrequency, acoustic, and optics data lines, complemented by devices for secret coding and data verification;
c) homeostasis supporting system [79], providing autonomous operation of devices for energy supply, time clocking, sensors operability control, and space positioning;
d) modules for decision making and work planning;
e) set of actuators.

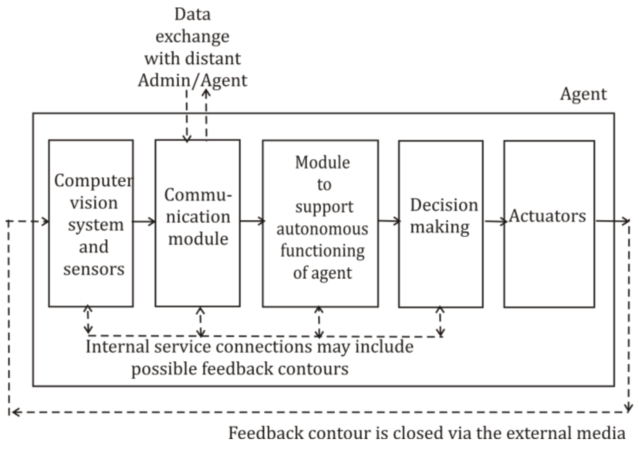

Data exchange with distant Admin/Agent
Agent
Computer vision system and sensors
Commu-nication module
Module to support autonomous functioning of agent
Decision making
Actuators
Internal service connections may include possible feedback contours
Feedback contour is closed via the external media

Figure 9. A highly simplified scheme of a robotic agent supposes the dialog exchange of messages with the distant administrator or other agent via the specialized communication module.

Principally, one of main properties of agent is to follow the principle of the so-called homeostasis in biological systems and to support autonomous work of a robot by self-sustaining of appropriate bands of "vital" parameters, like alive systems support concentration of oxygen and glucose in blood using the cyclic principle of work and control contours with feedback. Using internal subsystems, the agent executes received tasks and instructions [79], what somehow changes the external media. Agent sequentially monitors these changes by its sensors and repeats the work cycle, thus realizing closed contours of control with the feedback obtained via the external media. Such cycles can involve time bands varying at least from nanoseconds up to months, what needs to use correlated time variables. That is why the universal communication module of agent and verification procedures should model the detailed structure of parameters, and it can`t be limited by several verification templates.

The receiving messages agent is to take into account instructions transferred by the administrator and requests from other agents, what may quickly change states of the homeostasis contours. Only after this the agent is to activate decision-making procedures, which may cause series of actions and need various run time. As embedded modules of agent suppose somehow optimized mode of work, the realistic way is to run verification procedure after the completing of the next work cycle [59].

If some agent receives the initiating signal in a new phrase, the dialog scheme of two agents can include a vast spectrum of procedures and AI algorithms as follows:

- to check the received initial parole in the list of fixed and additionally generated ones;
- to extract potentially danger data to be immediately send to homeostasis module,
- to check the presence of the declared sender name and its authority in the AGA function, describing allowed classes of actions for the contacting agent,
- to verify the presence of all declared content parts in the received phrase;
- to compare the requested by external agent set of vocabularies and the allowed one;
- either to begin the new dialog, or to continue the already opened one; if necessary, to send denial reply;
- to send content of the received phrase to the decision-maker and to wait its reply;
- to work out the reply phrase and to assign to it a one-time random hash value, using the reserve list of hash values preliminary filled in by quasi random numbers;
- to add new entry into the LL of incoming messages to save the history of joint work;
- to analyze the reason of errors in the dialog and to choose adequate actions.

The proposed further scheme of dialog communications, combining AGA control of vocabularies and the binary calculation of total check coefficients, bases on **Tables 3, 5** and is given in **Figure 10**. During the dialog exchange of messages between agents within the same MAS we don`t make the difference between Administrator and Agent of MAS, although the first one should have additional data channels. All phrases $\boldsymbol{W}_{ij}$ are subdivided into initiating dialog requests and replies to them.

According to **Table 5,** all phrases in the request and reply messages include the header with obligatory several fields and content parts. Indexed values w_{ij} define some sampling of words, chosen from the corresponding set of vocabularies by the communication module and the decision maker. External communication module of administrator and agents adds hash value, which can be either fixed initial access password $w_{1,2}$, or further assigned quasi random hash value $w_{1,3}$=h_{ij}, which for better reliability should be obtained from cryptographic hash functions or QRNG [69]. In **Figure 10** it is supposed, that the communication session is a new one, and Sender of the phrase initiates the dialog, so that the first field "Format" in the header of phrase should contain initiation word w_1^K ="Hi". It activates further the input of initial access parole $w_{1,2}^K$ written in the vocabulary $V_{1,2}^K$. This set (or list) of fixed passwords is intended for the new sessions initiation only, as during the current communication session interacting agents should add one-time hash values taken from the preliminary generated list L_{hash}. Other components of the header of the phrase $\boldsymbol{W}_{\boldsymbol{11}} = \{ w_{1,1}^K,\ w_{1,2}^K,$ ….} are the triples $(w_{1,4}^K, w_{2,4}^K, w_{3,4}^K)$ and $(w_{1,5}^K, w_{2,5}^K, w_{3,5}^K)$, describing all given Addressers and Senders. Such 8-bit triple can support 16 777 216 agents.

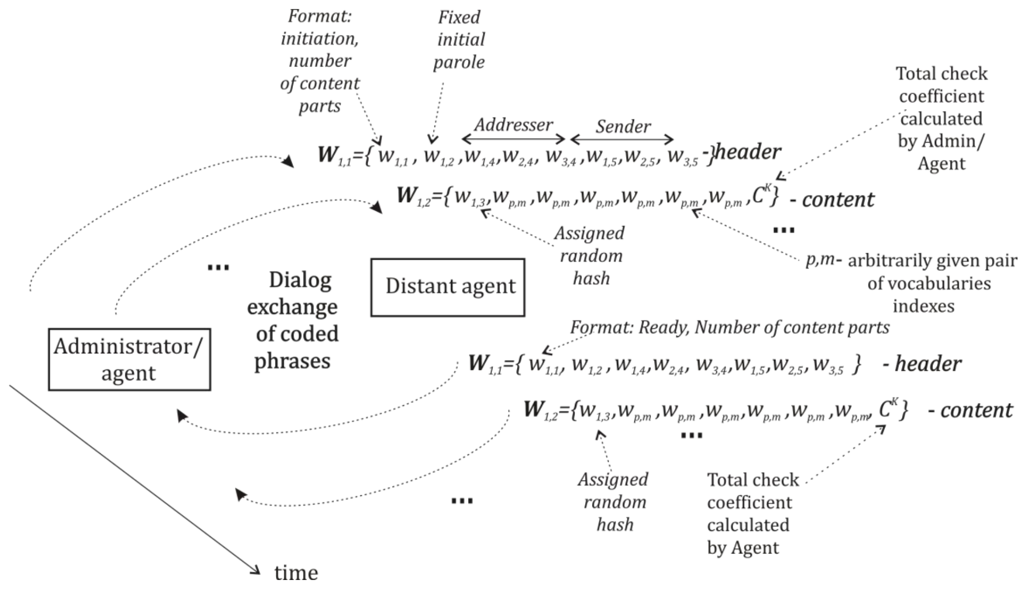

Format: initiation, number of content parts
Fixed initial parole
Addresser
Sender
Total check coefficient calculated by Admin/ Agent
$W_{1,1}=\{w_{1,1}, w_{1,2}, w_{1,4}, w_{2,4}, w_{3,4}, w_{1,5}, w_{2,5}, w_{3,5}\}$ - header
$W_{1,2}=\{w_{1,3}, w_{p,m}, w_{p,m}, w_{p,m}, w_{p,m}, w_{p,m}, w_{p,m}, C^K\}$ - content
Assigned random hash
p,m- arbitrarily given pair of vocabularies indexes
Dialog exchange of coded phrases
Distant agent
Administrator/ agent
Format: Ready, Number of content parts
$W_{1,1}=\{w_{1,1}, w_{1,2}, w_{1,4}, w_{2,4}, w_{3,4}, w_{1,5}, w_{2,5}, w_{3,5}\}$ - header
$W_{1,2}=\{w_{1,3}, w_{p,m}, w_{p,m}, w_{p,m}, w_{p,m}, w_{p,m}, w_{p,m}, C^K\}$ - content
Assigned random hash
Total check coefficient calculated by Agent
time

Figure 10. Principal scheme of dialog messages exchange between Sender (Administrator or Agent) and Addresser (distant Agent). Format of requests and replies corresponds to **Table 4**.

The phrase procession scheme shown in **Figure 10** corresponds to **Algorithm 1,** which is given in **Appendix A** due to its length. It describes the overall scheme of formal checks of initiating words, passwords, check coefficients, search of the header and content parts, and the waiting of reply in the dialog mode.

Table 6 demonstrates the numerical example of the phrase**,** disclosing the scheme to find correct content parts in the buffer using the header part. As besides the correct content part the receiving buffer of agent potentially can contain by a mistake some excessive illegally sent parts or ones, taken from someone else's message. Its analysis is based on simple summation of mapped truth levels and the comparison of total check coefficients, obtained by Addresser and declared by Sender.

Table 6. The example of the model phrase to clarify questionable codes of two words $w^K_{v,m}$ in actual vocabulary $V^K_{v,m}$. Correct phrase consists of the header and content.

Header:	$w^K_{1,1}$	$w^K_{1,2}$	$w^K_{1,4}$	$w^K_{2,4}$	$w^K_{3,4}$	$w^K_{1,5}$	$w^K_{2,5}$	$w^K_{3,5}$
Mapping to natur. numbers code	Hi= 2	Fixed parole of adresser=131	Addres-ser name1= 0	Addres-ser name2= 0	Addres-ser name3= 3	Sender name1= 0	Sender name2= 0	Sender name3= 2
Content part 1:	$w^K_{1,3}$	$w_{1,8}$	$w_{1,9}$	$w_{2,9}$	$w_{3,9}$	$w_{1,15}$	$w_{1,3}$	$C^{N/K}=\sum w_{v,m}$
Mapping to natur. numbers code	Assigned hash value =209	Action= Show K-Code for word in $V^N_{v,m}$=196	Num-ber of object= v= 3	Num-ber of object= m= 12	Num-ber of object= $w^N_{v,m}$ =6	Questionable code name $w^K_{v,m}$= 221	Hash for access to vocabularies group, including $V^K_{v,m}$ =147	138+ 209+196+3+ 12+6+221+ 147 =932 (overflow: >255); 932- 3x255=167

The simple scheme to select correct content part of the phrase in the buffer memory, containing several ones, is given in **Algorithm 2.**

Algorithm 2. The procedure to find correct content part, corresponding to the given header, basing on check coefficients.

Input:	$\{ w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}\}$,	← Header
	$\{ w^K_{1,3}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, C^{N/K}\}$,	← Content parts
	………..	
	(up to 30 parts in the buffer)	
Step	**Procedure**	**Example from Table 6**
1.	Find header part of phrase in the buffer memory	-
2.	Calculate with 8 bit overflow $C^{N/K}_{header}$ for the header	$C^{N/K}$ =138 (< 255)
3.	Find nearest content part	-
4.	Calculate with 8 bit overflow $C^{N/K}_{cont}$ for the content part	209+196+3+12+6+221+147= 794(overflow: >255)= = 794-3x255=29;
5.	Calculate with 8 bit overflow: $C^{N/K}_{header} + C^{N/K}_{cont}$	138+29=167 (< 255)
6.	Compare, if the calculated result is equal to the declared one in the last field of the content part:	-
	$C^{N/K}_{header} + C^{N/K}_{cont} = C^{*N/K}_{header} + C^{*N/K}_{cont}$;	167=167
	If yes, then write header and content part in the buffer;	yes
	other, then find and process next content part.	
Output:	→ Correct pair of header - content part of phrase is in the buffer	

The further procession of the received phrase $\boldsymbol{W}_{i,j}$ should include the check for compatibility of allowed tasks and vocabularies, mentioned in the request. The detailed analysis of this vast topic is out of this paper, only several remarks are to be given. The aim here is to detect incorrect content messages and to limit the access of robots to currently not allowed topics and knowledge. That needs to design special AGA function, which in general case should take into account the current state of the scene of action and the dynamics of its development. Respectively, this procedure needs to additionally estimate such control parameters as e.g., the level of danger d, trust level tl, and level of available resources r. The multi-grade parameter "danger level" may be varied by an administrator, computer vision system, and sensors, what differs it from the "trust level" one, which supposes long enough monitoring of behavior of objects in the scene of action. This variable together with the trust level seems to use the simplified discrete structure of AGA truth levels with 5-10 grades, but the estimation of them may need probabilistic calculations. Resources data (like batteries charge voltage) can be united in one final logic variable, but complex planning algorithms may be needed to estimate the possibility to carry out the task. Thus, the principle of multiparametric AGA logic phrase procession is simple enough, but realistic robotics needs to add many expert data. Some of them are preferably to be written as templates for verification and should be protected from illegal modifications by the adaptation of earlier proposed schemes of AGA-based LL [52,53].

4. Results: LL Adaptation for Dialog and Verification Procedures

If verification and data restoration procedures are not supposed to be often held, massive dubbing of all incoming and sent messages by copying or AGA-based LL [52,53] seems to be appropriate mainly for administrator`s archiving, possessing substantial resources. Respectively, it seems reasonable to use shortened adaptations of AGA version of LL especially designed for messages aggregation and completed work history documenting. Such confidential data structure should contain brief information concerning earlier carried out tasks which may be confidential enough. The task to avoid illegal data modifications undertakes the problem of often repeated values of words $w^K_{v,m}$, necessary for typical robotic tasks, where even the space localization of a robot prompts most probable combinations of used in dialog words. LL schemes can additionally include total check coefficients $C^{K/N}$ (using summation of mapped truth levels) and ones C^{KXR} (using binary XOR operations). Such total coefficients for parts and whole phrases complicate the external prediction of instructions for frequently repeated tasks. Also one can periodically change the coding for truth levels representation, and for situations with the high risk of illegal activity of cheaters one can exploit standard NIST techniques of secret coding and AGA-based analog of the well-known one-time cipher pad scheme [18].

The problem to be solved here is that AGA versions of LL [52,53] initially suppose equal dimension of all entries. It determines the need to use short enough 8-byte phrase parts, easy for procession by mass platforms with 8-bit microcontrollers with embedded arithmetic summation and logic XOR operations. The scheme for such shortened LL is proposed further for the documenting of tasks, transmitted by phrases composed of g parts and including check coefficients $\mathrm{C}_{1,m-1}^{K,header}, \mathrm{C}_{2,m-1}^{K,content}, \dots, \mathrm{C}_{g,m-1}^{K,total}$. The basic version of LL is citated for comparison in **Table 7** (**a**) with more convenient arrangement of indexes, as in AGA the order of succession is not substantial for input variables. Initially defined model corresponds to the truth table where the newcomer entry is $\mathbf{e}_m = (\mathrm{e}_{1,m}, \dots, \mathrm{e}_{n,m})$, the previous entry is $\mathbf{e}_{m-1} = (\mathrm{e}_{1,m-1}, \dots, \mathrm{e}_{n,m-1})$, and approving sets of hash values are $\mathbf{h}_m = (\mathrm{h}_{1,m}, \dots, \mathrm{h}_{Q,m})$ and $\mathbf{h}_{m-1} = (\mathrm{h}_{1,m-1}, \dots, \mathrm{h}_{Q,m-1})$. Time moments stamps for entries are t_m and t_{m-1}.

Then in order to raise the variability of data traffic one can enlarge the number of total check coefficients $C_{i,j}^K$, used in content parts of phrases. Note that coefficients $C_{i,j}^K$ contain randomly assigned hash values besides predictable work parameters. Such version of LL in **Table 7 (b**) is designated for dialog communications of agents, if intensive replication of instructions or data is undesirable due to unreliable network, weak hardware, financial motives, or commercial and technology secrets. In this case, the mass approving of entries by external network nodes like in blockchain scheme [66–68] is problematic, and data approving is to be based on phrase parameters, fixed by the partner agent or by reserve copies in a pair of loyal or trusted agents.

Table 7. a) The base version of AGA linked list and its truth table is defined by function $\mathrm{h}^{(m,1)} = \mathrm{F}_{ldg}(\mathrm{m}, \mathrm{t}, \mathrm{e}_m, \mathrm{e}_{m-1}, \mathrm{h}_m, \mathrm{h}_{m-1})$. **b)** Proposed shortened version of LL for documenting of task parameters, transmitted by phrases, consisting of g parts and containing only check coefficients $\mathrm{C}_{1,m-1}^{K,header}, \mathrm{C}_{2,m-1}^{K,content}, \dots, \mathrm{C}_{g,m-1}^{K,total}$. Quasi random hash values $\mathrm{h}_1^{(1,1)}, \dots, \mathrm{h}_1^{(m,1)}$ approve corresponding entries.

a) Number of input variables: 2×(1+p+q), p-number of words in entry, q- maximal number of verifying participants

			Input						variables					Output variable
Common counters		*Previous Entry (8 bytes)*			*Verifying hash*			*Last Entry (8 bytes)*			*Verifying hash*			*Output hash*
m	**t**	$\mathbf{e}_{1,t-1}$	**…**	$\mathbf{e}_{p,t-1}$	$\mathbf{h}_{1,t-1}$	**…**	$\mathbf{h}_{q,t-1}$	$\mathbf{e}_{1,t}$	**…**	$\mathbf{e}_{p,t}$	$\mathbf{h}_{1,t}$	…	$\mathbf{h}_{q,t}$	$\mathbf{h}^{(m,1)}$
1	t_1	$e_{1,0}$		$e_{p,0}$	$h_{1,0}$	…	$h_{1,0}$	$e_{1,1}$	…	$e_{p,1}$	$h_{1,1}$	…	$h_{Q,1}$	$h_1^{(1,1)}$
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
m -1	t_{m-1}	$e_{1,m-2}$	…	$e_{p,m-2}$	$h_{1,m-2}$	…	$h_{Q,m-2}$	$e_{1,m-1}$	…	$e_{p,m-1}$	$h_{1,m-1}$	…	$h_{Q,m-1}$	$h_1^{(m-1,1)}$
m	t_m	$e_{1,m-1}$	…	$e_{p,m-1}$	$h_{1,m-1}$	…	$h_{Q,m-1}$	$e_{1,m}$	…	$e_{p,m}$	$h_{1,m}$	…	$h_{Q,m}$	$h_1^{(m,1)}$

b) Number of input variables: 2×(2+g), g-number of parts in the phrase, q- maximal number of verifying participants.

Common counters		*Previous Entry (8 bytes)*				*Verifying hash*	*Last Entry (8 bytes)*				*Verifying hash*	*Output hash*
		Input					**variables**					**Output variable**
m	**t**	$\boldsymbol{C_{1,m-1}^{K,head}}$	$\boldsymbol{C_{2,m-1}^{K,cont}}$	…	$\boldsymbol{C_{g,m-1}^{K,total}}$	$\mathbf{h}_{\mathbf{i},\boldsymbol{m-1}}$	$\boldsymbol{C_{1,m}^{K,head}}$	$\boldsymbol{C_{2,m}^{K,cont}}$	…	$\boldsymbol{C_{g,m}^{K,total}}$	$\mathbf{h}_{\mathbf{i},\boldsymbol{m}}$	$\mathbf{h}^{(\mathbf{m},\mathbf{1})}$
1	t_1	$C_{1,1}$	$C_{2,1}$	…	$C_{g,1}$	$h_{1,1}$	$C_{1,2}$	$C_{2,2}$	…	$C_{g,2}$	$h_{1,2}$	$h_1^{(1,1)}$
…	…	…	…	…	…	…	…	…	…	…	…	…
m-1	t_{m-1}	$C_{1,m-2}$	$C_{1,m-2}$	…	$C_{g,m-2}$	$h_{m-1,m-2}$	$C_{1,m-1}$	$C_{1,m-1}$	…	$C_{g,m-1}$	$h_{m-1,m-1}$	$h_1^{(m-1,1)}$
m	t_m	$C_{1,m-1}$	$C_{2,m-1}$	…	$C_{g,m-1}$	$h_{m,m-1}$	$C_{1,m}$	$C_{2,m}$	…	$C_{g,m}$	$h_{m,m}$	$h_1^{(m,1)}$

Exp. (8) demonstrates the formal logic notation of truth table given in **Table 7 (b)**.

$$\begin{aligned}
&h^{(out)} = \\
&h^{(1,1)} \star X_m(1,1) \star X_t(t_1,t_1) \star X_{C,1,1}(C_{1,1},C_{1,1}) \star \ldots \star X_{C,g,1}(C_{g,1},C_{g,1}) \star \\
&X_{h,1,1}(h_{1,1},h_{1,1}) \star X_{C,1,2}(C_{1,2},C_{1,2}) \star \ldots \star X_{C,g,2}(C_{g,2},C_{g,2}) \star\ X_{h,1,2}(h_{1,2},h_{1,2}) + \cdots + \\
&h^{(m-1,1)}\ \star X_m(m-1,m-1) \star X_t(t_{m-1},t_{m-1}) \star X_{C,1,m-2}(C_{1,m-2},Ce_{1,m-2}) \star \ldots \star \\
&X_{C,g,m-2}(C_{g,m-2},C_{g,m-2}) \star X_{h,1,m-2}(h_{1,m-2},h_{1,m-2}) \star X_{C,1,m-1}(C_{1,m-1},C_{1,m-1}) \star \\
&\ldots \star X_{C,g,m-1}(C_{g,m-1},C_{g,m-1}) \star X_{h,1,m-1}(C_{1,m-1},C_{1,m-1}) +\ \ h^{(m,1)}\ \star X_m(m,m) \star \\
&X_t(t_m,t_m) \star X_{C,1,m-1}(C_{1,m-1},C_{1,m-1}) \star \ldots \star X_{C,g,m-1}(C_{g,m-1},C_{g,m-1}) \star \\
&X_{h,m,m-1}(h_{m,m-1},h_{m,m-1}) \star\ X_{C,1,m}(C_{1,m},C_{1,m}) \star \ldots \star X_{C,g,m}(C_{g,m},C_{g,m}) \star \\
&X_{h,m,m}(h_{m,m},h_{m,m}). \qquad (8)
\end{aligned}$$

However, for simple robots and IoT with primarily non-confidential data one can further shorten the volume of dialog data and simplify the set of instructions by means of adaptation of LL_{wrd}, providing direct output of requested group of words.

Adapted AGA function LL_{wrd}^{ad} can be written by exp. (9)

$$w_{v,m}^K = F_{wrd}(w_{v,m}^N, v, m, h_{v,m}^{(m,0)}, h_{v,m}) \qquad (9)$$

which includes the rearranged parameters $w_{v,m}^K$ and $h_{v,m}^{(m,0)}$ and is discussed in sec. 6, **Table 10**.As in fact $w_{v,m}^K$ are also quasi random hash values taken from one QRNG, then one can quite freely permute them instead of quasi random hash values $h_{v,m}^{(m,0)}$ earlier assigned by

Addresser. Parole $h_{v,m}$ or Sender`s approving hash is considered as a randomly chosen key for the access to data in another agent, but in more exhausted schemes one can define various paroles for different groups of agents. Microassembler modelling for the function exp.(9) of the simplified LL is presented in sec.6.

One should note that schemes given above doesn`t exclude other variants. As the structure of a phrase was chosen consisting of equal 8-byte parts, one can e.g., use theversion of LL sequentially mixing entries, formed by the header and the 1st content part, the 1st content part and the 2d content part, etc.. However, the detailed discussion of it needs to discuss possible tasks and efficiency criteria, what is out of this paper.

Proposed above schemes give the formal logic base for autonomous selective documenting of agent`s parameters in local distributed data storages.

However, verification algorithms should be supplemented by human checks.

5. Results: Fragmentation of AGA Function into Diagrams of Logic States for Verification

The most obvious way to held selective verification procedure is to map the set of critical data from the holistic logic model onto low-dimensional 1D, 2D or 3D charts or diagrams of logic parameters, which are convenient for quick visual checks by human experts, who are the final authority. For verification e.g., in the position-based cryptography and route passing tasks [22], one can use time diagrams of visits to check points, 2D space maps of alternatively visited checkpoints or cargo terminals, time diagrams of the execution of the quantum protocol and verification procedures [52,53]. But the design of such procedures for AGA needs to give several definitions.

Definition 5. Fragmented mapping function (FMF) $F(x_l, \dots, x_m)$ taken within AGA is the result of the injective mapping $F(x_1, \dots, x_n) \to F(x_l, \dots, x_m)$, where:

1) $n, l, m \in N = \{0,1, \dots, k-1\}$, $m \geq l$, and $n-1 \geq m-l$, (i.e. the mapping shortens the number of initially taken input variables);

2) truth tables of the mapping function $F(x_1, \dots, x_n)$ and the mapped one $F(x_l, \dots, x_m)$ have:

a) identical data in columns for coinciding input variables $x_l, \dots, x_m$;

b) identical logic constants C_k in columns for output variables $F(x_l, \dots, x_m)$ and $F(x_1, \dots, x_n)$.

*Consequence.*FMF can be defined for any sample of initially given input variables. For its practical formation one should simply strike out further in **Table 8** all columns for unused input variables, not included into the necessary set $\mathrm{x}_l, \dots, \mathrm{x}_m$.

Definition 6. The fragmented map M of the fragmented mapping function $F(x_l, \dots, x_m)$ is the group of points, mapping all given samples of input variables $x_l, \dots, x_m$ from the truth table onto the arbitrarily chosen set of:

1) logic constants $[C_{k1}, .., C_{k2}]$, where $C_{k1,k2} \in \{1, .., k-1\}$, or

2) output variables $x_j, j \in \{l, .., m\}$.

Note. The fragmented map M shows the location of all objects or events with the chosen class given by the constant C_k; the limiting band for logic constants or input variables corresponds to critical parameters given by the task.

Definition 7. Fragmented mapping function (FMF) is named the verification function (VF), if it belongs to the list of formal criteria L_e, formed and checked by the knowledge expert.

Given above definitions are illustrated further by **Table 8**, demonstrating the example of formation of FMF $F(x_3, x_4)$ by striking out unnecessary variables x_1, x_2, x_5 from the initially given truth table. We suppose that coordinates x_3, x_4 correspond to space ot parameters x, y, which can be the correlated ones, if e.g., the FMF $F(x_3, x_4)$ shows the location of visited by robot objects or checkpoints. As correlated space and time variables (see sec.2.3) are considered in AGA model as different logic variables, they also can be selected for checks. Principally, any 2D or 3D mapping diagrams like $F(x,t)$, $F(x,y)$, or $F(x,y,t)$, given in the band $[0, k-1]$, can be considered as verification functions if adopted by the knowledge expert. 2D logic diagram in fact resembles group point images, known in star navigation systems [81], which can be checked directly on the display or may be additionally processed.

Table 8. Example of FMF $F(x_3, x_4)$, formed by striking out unnecessary variables in the holistic truth table. Mapping function is $F(x_1, \ldots, x_5)$, mapped one is $F(x_3, x_4)$, restriction is $C_k = \{5{,}6\}$..

Input	variables				Output variable
~~x_1~~	~~x_2~~	x_3	x_4	~~x_5~~	$F(x_1, \ldots, x_n)$
~~…~~	~~…~~	1	1	~~…~~	5
~~…~~	~~…~~	1	3	~~…~~	6
~~…~~	~~…~~	1	8	~~…~~	5
~~…~~	~~…~~	1	4	~~…~~	6
~~…~~	~~…~~	2	3	~~…~~	5
~~…~~	~~…~~	2	4	~~…~~	6
~~…~~	~~…~~	2	5	~~…~~	5
~~…~~	~~…~~	7	3	~~…~~	6

It is necessary to accent the fact, that FMF can be formed arbitrarily for visual checks, but the reverse aggregation into a new AGA functions needs to monitor the values of all the variables, for which FMF was initially defined. The reason is that according to basic definitions of AGA [51], any notation like $F(x_i, x_j) = F(0,0,\ldots,0) * X_i(a_i, b_i) * X_j(a_j, b_j)$ always can be extended to the function expression $F(x_i, x_j, x_k) = F(0,0,\ldots,0) * X_i(a_i, b_i) * X_j(a_j, b_j) * X_k(0, K-1)$, where the band from 0 up to $K-1$ will exceed the ranks of earlier used model.

Reverse equivalent representation from truth levels codes $w_{v,m}^{K}$ back into natural numbers ones $w_{v,m}^{N}$ can help to receive more visual and convenient groups of parameters, and logic diagrams taken with various limitations are expected to be useful for more complicated data analysis during the debugging of mobile agents. Namely the natural numbers coding of words can help to form convenient groups of data for further planned verification procedures. **Figure 11** shows the idea of such logic point group images, obtained for 2D logic diagrams taken from **Table 8** with restriction $C_k = \{5{,}6\}$, which correspond to some vocabularies $V_{v,m}^{K}$. For visual checks by human experts, it can be useful to connect additionally points of the logic diagram by lines. Even the simplest 1D functions $F(x_i) = F(0,0,\ldots,0) * X_i(a_i, b_i)$, $i \in [0,1,\ldots,k-1]$ also corresponds to the given above definition of FMF and may describe numbers of agents and their licenses, obtained parameters, or passed distances.

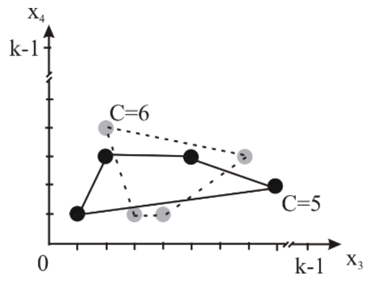


Figure 11. 2D logic diagram displays only events classified by logic constants 5 and 6 for the AGA function given in Table 2 and limited by the criteria $C_k = \{5{,}6\}$. Close constants should be preliminary reserved in the natural numbers code to obtain convenient diagrams for close classes of objects.

Given above Defs. (5÷7) describe FMF via truth tables, but basing on AGA template matching and distributed linked list schemes [53,54] one can directly use equivalent representations of the truth table by logic expressions given in sec.2. For AGA logic diagram truth levels always have fixed values and only the number of involved logic constants and their relative location in the diagram may differ. Consequently, AGA-based logic diagrams are the more simple case of check images in contrast to, e.g., contours in stars navigation systems [81].

Thus, FMF scheme enlarges the spectrum of possibilities for human verification and small-scale local memory storages. Due to the fact, that LL is AGA function, it can be used not only as a holistic data structure, but also as a data source for FMF dubbed in a local distributed memory storage. However, all these variants need to discuss hardware requirements. Here the goal is to demonstrate the possibility to realize proposed methods by means of the simplest 8-bit platform, corresponding to IoT level.

6. Results: Communication Module and Microassembler Software for the Verification

The possibility to process the proposed vocabularies structure and the adapted LL scheme is demonstrated by means of the microassembler code and 8-bit circuit board, which was already used in [53,54,59]. Such choice is determined by three factors, and the first of them is to test new AI procedures for IoT devices level, where one don`t use special complicated AI software shells, NN, and specific microprocessors. The second reason is the design of the communication module, which should unload the decision making module in agent. The third reason is that the communication module of agent should protect as much as possible the system from illegal codes and data, what traditionally infers low-level programming based on detailed understanding of memory resources.

Now proposed programs complement a dozen of microassembler subroutines for LL scheme, which were earlier presented in [53,54] and disclosed basic necessary functions of AGA for agent operation and its interface with PC. Given microassembler software was tested in the so-called dual-chip module [53,54,59], based on two 24 MHz ATMEL microcontrollers MCS-51. Nevertheless, the enlarged set of procedures for dialog communications needs to propose the adequate solution for the communications module, which is shown in **Figure 12** and is formed by the pair of sequentially connected microcontroller dual-chip cascades from [53,54,59], modified by digital 8-bit feedback bus. It`s role is to provide flexible forward/backward data exchange berweeh modules, controlled by means of several control pins. The bidirectional transfer of data is supposed to be used as between "upper" and "lower" dual-chip modules, as for other peripheral modules. Like in [53,54] the conjugation with the peripheral devices is to use bus registers, triggered on the leading edge.

The advantage of such device is the pair of conjugated microcontrollers connected to common data and address buses, what provides the possibility to use them in parallel or to exploit the doubled set of operators (500-600 ones per MCS-51) for the joint volume of information written in SRAM and ROM. Such structure simplifies the programming of AGAprocedures and helps to separate data acquisition and procession.

Besides this, it gives more possibilities to combine connections of pins and to add new trigger registers for alternative memory chips connected to the common data bus. Two types of integrated memory SRAM and ROM chips are to separate fixed and optional parameters. Such cascaded scheme with microcontrollers currently had 1 MB SRAM and 512 KB EEPROM external memory chips, which were connected to common data buses and provided free enough data resending and memory planning for laboratory experiments. That is the reason to use dual-chip schemes instead of commercial kits like Arduino [82], although they can use greater work frequencies.

For simplicity we don`t discuss here possible radiofrequency and optoelectronic data channels, and the specifics to connect them to the dual- chip scheme. In any way for such designs one needs to describe transmitters/receivers by some vocabularies and to reserve memory resources and free pins in microcontrollers for control signals. These steps can somewhat enlarge the overall time response due to the increased number of control pins and trigger registers, but for the proposed scheme of vocabularies the only requirement is to use equal 8-byte parts of phrases and to coordinate all equivalent vocabularies. As in [54], trigger registers Rg1, 2, and 3 are intended for SRAM addressing, where pins $\overline{CE}$, $\overline{OE}$, $\overline{WE}$ control read/write process. Banks P0 in MCI and MCII are used for data exchange with SRAM&ROM. Port P3 in MCII is used for clocking procedures, where direct and inverted signals of pins P3.0-P3.4 control read/write procedures. The first of dual-chip schemes is to interact with data receivers/transmitters and is supposed to transmit received data to SRAM into the second dual-chip module. Respectively, the procession of fixed and assigned paroles together with verification procedures was considered for the second one.

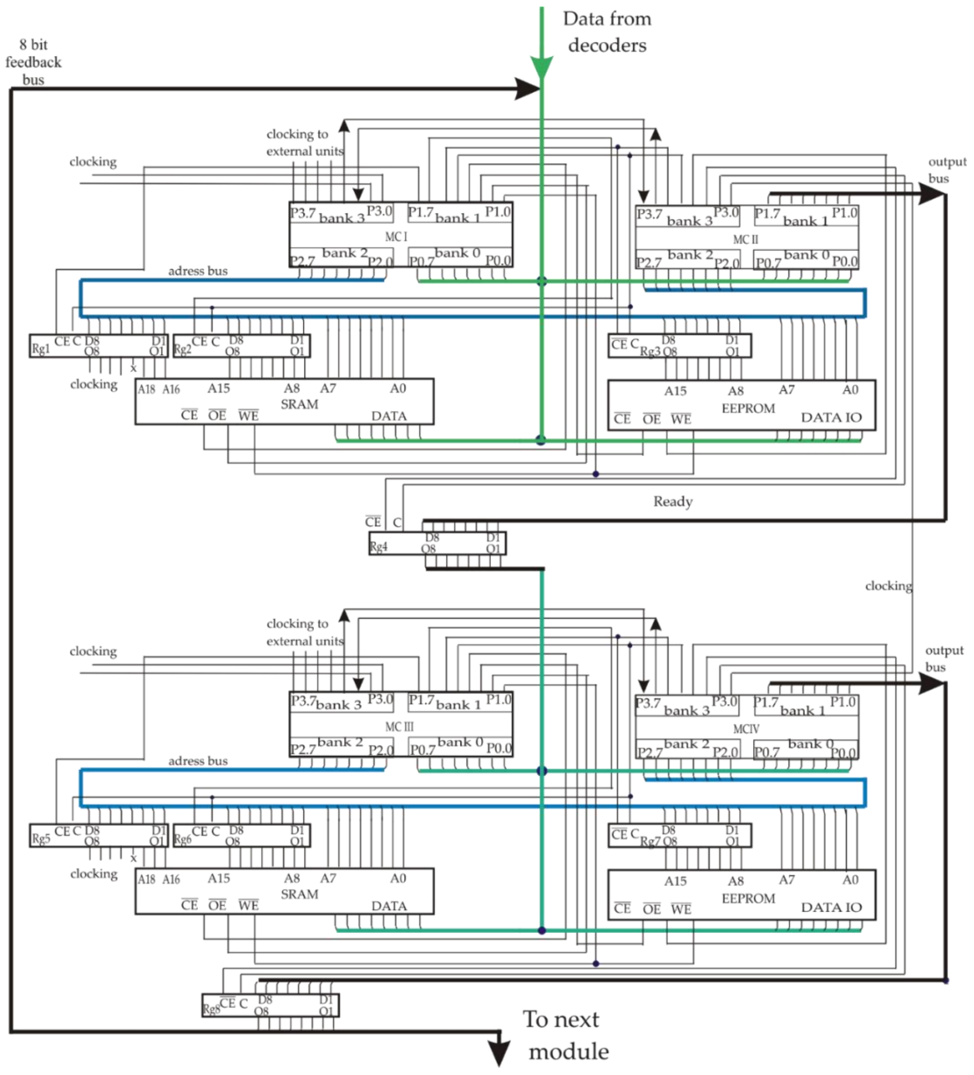


Figure 12. Two dual-chip modules are necessary to process full-scale dialog phrases, based on the proposed vocabularies structure.

In order to use the same EEPROM in dual-chip circuit boards as in [53,54], parameters v, m for vocabularies $V^K_{v,m}$ were taken $v = 16$, m=16, the number of root vocabularies q was also taken to be 16, what gives 4096 different words $w^K_{v,m}$; this responds to multiplexing of 16 root words into 16 subsets and once again multiplexing into 16 next ones. One should especially note, that although every vocabulary contains up to 16 different words, memory allocation tables provides natural number codes and corresponding truth levels codes in the band {1÷255}, where zero means the not used parameter**.** For advanced systems, Addresser may "demand" additional credentials and ask additional questions to Sender for the full access to the vocabularies structure.However, for simple robots in IoT one can shorten the number of dialog phrases and simplify the set of instructions by means of the adaptation of LL_{wrd}, providing direct output of requested group of words. As various specialized agents may be allowed to use only selected groups of vocabularies, they should keep in memory the preliminary given table of access paroles or fixed hashes $\mathrm{h}_{\mathrm{v,m}}$ and approval hashes $\mathrm{h}^{(\mathrm{m},0)}$, which principally may be chosen different for various groups of robots.

In the present paper the main task of modeling was to show the possibility to use the proposed vocabulary structure and adapted LL scheme for the verification of confidential enough data, where one avoids direct full data transfer, and Sender of phrase only checks the correctness of actual for him words by means of correct template hidden in LL_{wrd}. Variant of memory allocation table for the adapted version of the linked list LL_{wrd} is shown in **Table 10;** it resembles schemes used in [54], but here both SRAM and EEPROM chips were used. This adapted AGA function LL_{wrd} follows exp.(10)

$w^K_{v,m} = F_{wrd}(w^N_{v,m}, v, m, h^{(m,0)}_{v,m}, h_{v,m})$ taken with rearranged parameters $w^K_{v,m}$ and $h^{(m,0)}_{v,m}$. Due to the fact, that $w^K_{v,m}$ are also quasi random hash values, one can rearrange it with another quasi random hash values $h^{(m,0)}_{v,m}$ assigned by Addresser, what corresponds to definitions of LL [53,54].

Table 10. Memory allocation table used for modelling of LL_{wrd}. .

N	Word code in natural numbers, $w^N_{v,m}$={1÷16}	Number of row in vocabulary matrix, v = {1÷16}	Number of column in vocabulary matrix, m={1÷16}	Addresser approving hash, $h^{(m,0)}_{v,m,}$={1÷256}	Sender approving hash, $h_{v,m}$ = 1÷256}	Word code in truth levels, $w^K_{v,m}$ ={1÷256}
#A18 – A16:	#000b	#000b	#000b	#000b	#000b	#000b
#SB (A15-A8):	#109	#108	#107	#106	#105	#104
#LB:	#0÷255	#0÷255	#0÷255	#0÷255	#0÷255	#0÷255

Further modelling is demonstrated by the microassembler subroutine LLWRD shown in **Algorithm 3.** It refers to the the adapted version of the linked list LL_{wrd} proposed above in sec. 4 and described in **Table 10.** It`s role is to approve correctness of the truth levels code for some of words $w^K_{v,m}$, according to the request sent by an external agent Sender, which should declare several passwords to Addresser and to check the correctness of his version of truth levels code for actual coded words, using template data written in LL_{wrd} in Addresser.

The second presented program fragment of the subroutine CHKPHR is shown in **Algorithm 4** and refers to the formal procession of the header part of the phrase by Addresser**,** as the consideration of content part needs to analyze the vast structure of typical agent`s tasks.

Both procedures are supposed to be carried out in the communication module of agent, which is to carry out preliminary procession of messages, as the full scale AI modeling is the prerogative of the decision-making module.

Microassembler subroutine LLWRD is given in **Algorithm 3** and continues earlier designed set of software programs presented in for the dual–chip scheme and the base version of LL. It partially intersects with basic algorithms given as Tables 12-13-14 in [54], but there only SRAM was involved. Here EEPROM was used as the data storage for template (or correct) data, and SRAM was to keep data to be checked, which were received from Sender. EEPROM chip needs separate addressing to compute a set of non-zero product terms $\{pt_1,..., pt_m\}$ [53,54].The same #SB addresses (## 107÷103) were reserved for the LL_{wrd} in SRAM and EEPROM to shorten the re-switching of pins. Also, instead of calculation of the operator MIN in pairs $PT_m = MIN(h^{(m,1)}, pt_m)$ [54], more short program LLWRD directly assigns logic constant $h^{(m,1)}$ to the non-zero product term pt_m with the result saved in *#SB=#103.* **Algorithm 3** demonstrates simple adaptation of vocabularies structure to the LL scheme. If necessary, some more complicated versions can be used for dialog polling and data approving.

Algorithm 3. Subroutine LLWRD is to compute correct truth levels codes for the requested elements of vocabulary $V^K_{v,m}$ and $V^N_{v,m}$. Procedure is to be carried out in MCS-III of 2d dual-chip and uses its pins.

INPUT: $w^N_{v,m}$, v, $m, h_{v,m}$, $h^{(m,0)}_{v,m}$ ← Input variables

```
1  LLWRD:  CLR P1.4; prepare pin
2  MOV P2,#000b; Assign #A18-A16=#000b for SRAM
3  CLR P1.7; enable Rg1 by $\overline{CE}$
4  SETB P1.4; Rg1 writes #A15-A8=#000b
5  CLR P1.4;
6  SETB P1.7 ; lock Rg1
7  CYCNT: MOV R2,#255; Set counter of notations
8  CYCVAR: MOV R3,#5; counter of input vars
9  LOAT:MOV P2,#009b; set #SB at to Rg3/7 ROM
10 CLR P1.5; enable Rg3/7 by $\overline{CE}$
11 SETB P1.4; Rg3/7 writes #SB=#009b for a-template
12 CLR P1.4;
13 SETB P1.5; lock Rg3/7
14 MOV P2,R2;  set #A7-A0=#255 for at ROM
15 CLR P1.2; enable  ROM by $\overline{CE}$
16 CLR P3.1; enable ROM  output by $\overline{OE}$
17 REAT:MOV R7,P0; read at from ROM
18 SETB P3.1; disable ROM output by $\overline{CE}$
19 LOBT:  MOV P2,#109; load #SB  of bt in ROM
20 CLR P1.6; enable Rg3/7 by $\overline{CE}$
21 SETB P1.4; Rg3/7 writes #SB=#109 of b-template
22 CLR P1.4;
23 SETB P1.6; lock Rg3/7
24 MOV P2,#255;  set #LB=#255 for  ROM
25 CLR P1.2; enable chip ROM by $\overline{CE}$
26 CLR P3.1; $\overline{OE}$ enables ROM  output
27 REBT:MOV R6,P0; read b-template from ROM
28 SETB P3.1; disable ROM output
29 SETB P1.2; disable chip ROM
30 CLR P1.1; prepare pin for SRAM
31 LOAE: MOV P2,#0;  set #SB=0 in Rg1/5for SRAM
32 CLR P1.6; $\overline{CE}$enables Rg2
33 SETB P1.4; write #SB=#0 to Rg2
34 CLR P1.4
35 SETB P1.6; lock Rg2
36 MOV P2,R3 ; #LB counter of variables
37 CLR P1.3; $\overline{CE}$ enables SRAM
38 CLR P1.1;$\overline{OE}$ enables output of SRAM
39 REAE:MOV R5,P0; read a-external from SRAM
40 SETB P1.1; disable output of SRAM
41 SETB P1.3; disable SRAM
42 LITERAL: MOV A,R7; load a-template to calc Lit.
43 CLR C; prepare carry bit
44 SUBB A, R5;  at-ae=R7-R5
45 JC CAB; jump if carry bit C=1,i.e. a-ext. >a-templ.
46 AJMP PT0; Lit=0 and the whole pt=0
47 CAB: CLR C
48 MOV A,R6;  load bt  to A to calc Literal
49 CLR C; prepare carry bit
50 SUBB A,R5;  bt-ae=R6-R5
51 JC PT0;  jump PT0 if bit C=1, i.e. ae> bt
52  DEC R3; counter of input vriables
53 CJNE R3,#0,LOAT; process next variable
54  PT1: AJMP WRRES; write const in #SB=#003
55 PT0: MOV R0, #0; product term=#0
56 AJMP WRR0
57 WRRES: MOV P2,#4 ; set #SB=#004 to read Const
58 SETB P1.4; write #SB=#004 into Rg2/6
59 CLR P1.6; enable Rg2/6
60 CLR P1.4;
61 MOV P2,R3 ; assign #LB to read Const
62 CLR P1.3; $\overline{CE}$ enables chip SRAM
63 CLR P1.1; $\overline{OE}$ enables output of SRAM
64 RECNST:MOV R0,P0; write Const in R0
65 SETB P1.1; disable output of SRAM
66 SETB P1.6; disable Rg2/6
67 WRR0: MOV P2,#3; set #SB=#003 to write Const
68 CLR P1.6; enable RG2/6
69 SETB P1.4; write #SB=#003  to Rg2
70 CLR P1.4;
71 SETB P1.6; lock Rg2 with #003
72 MOV P2,R2; addressing of #LB for SRAM
73 MOV P0,R0; output next Const
74 CLR P1.3; $\overline{CE}$ enables chip SRAM
75 CLR P1.1; $\overline{OE}$ enables data  of SRAM
76 CLR P1.0; write Const in #SB=003, #LB=R2
```

```
SETB P1.0; $\overline{WE}$ disables write in SRAM
SETB P1.1; $\overline{OE}$ disables data  of SRAM
SETB P1.3; $\overline{OE}$ disables chip SRAM
DJNZ R2,#1,CYCVAR; process next notation
MAXPTS:MOV R1,#103; #SB103 for PTs
MOV R2,#255; counter of PTs
MOV P2,R1; addressing #SB=#103
CLR P1.6; enable Rg2 by $\overline{CE}$
SETB P1.4; write #SB=#103 to Rg2
CLR P1.4
SETB P1.6; lock Rg2
MOV P2,R2 ; #LB is the counter of PTs
CLR P1.3; enable SRAM by $\overline{CE}$
CLR P1.1;$\overline{OE}$ enables output of SRAM
MOV A,P0; read PT
SETB P1.1; disable output of SRAM
SETB P1.3; disable SRAM
DEC R2
NEXTPT:MOV P2,R2
CLR P1.3; enable SRAM by $\overline{CE}$
CLR P1.1; $\overline{OE}$ enables output of SRAM
MOV R7,P0; read  next PT
SETB P1.1; disable output of SRAM
SETB P1.3; disable SRAM
MOV R3,A; save value of Acc
CLR C
SUBB A,R7;
JNC MAX_N1
MOV R0,R7
MAX_N1:MOV R0,R3
DJNZ R2, NEXTPT
RETI
```

OUTPUT: R0 → $w^K_{v,m}$, truth levels code for the requested word from $V^K_{v,m}$ is written into the register R0

The fragment of the second microassembler program CHKPHR is to find correct parts in the buffer and for brevity is given fragmented. Input data buffer in SRAM of the 2d dual-chip uses the allocation memory table given in **Table 11.** Memory reserve fields are necessary to write intermediate calculations and results.

Table 11. SRAM buffer structure for received phrases 8-byte parts in the 2d dual-chip module**.** It can contain 30 written parts, obtained from the receiving module. #LB provides up to 256 possible code numbers for various words in a vocabulary.

	Received data buffer				**Reserve for intermediate calculations**	
N	**Part 1**	**Part 2**	**…**	**Part 30**	**Results 1**	**Results 2**
#A18 –A16:	#000b	000b	**…**	000b	000b	000b
#SB (A15-A8):	#111b	#111b	**…**	#111b	#111b	#111b
#LB:	#0-7	#8-15	**…**	#233-239	#240-247	#248-255

Further **Algorithm 4** demonstrates the initial fragment of the microassembler program CHKPHR to process the initiation word, to check input parole, to check addresser, to calculate total check coefficient summation, and to compare it with the declared value, i.e. it can determine correct header in the buffer.

Algorithm 4. The fragment of subroutine CHKPHR is to detect header part of phrase and to calculate its total check coefficient in the buffer data. Register R7 is the counter for 30 parts in the buffer segment, R6 is the counter of parts in the phrase, R5 - counter of #LB, R4 - counter of word values (limited by 16).

Input:	$\boldsymbol{W_h} = \{w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}\}$	←Header of the received phrase parts
	$\boldsymbol{W_c^*} = \{w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}\}$	←Error 8-byte part
	$\boldsymbol{W_c} = \{w^K_{1,3}, w^K_{1,8}, w^K_{2,8}, w^K_{33,s}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, C^{N/K}_{total}\}$	←Correct content part of phrase
	$w^K_{1,2} = 7 = Hi, read\ 2\ parts,$	←Words interpretation in the decoding table
	$w^K_{1,2} = 123 = Hi, read\ 3\ parts,$	
	$w^K_{1,2} = 209 = Hi, read\ 4\ parts$	

```
FINDHI: CLR P1.4; prepare pin
MOV P2,#00000111b; Assign #A18-A16=#000b
CLR P1.7; enable Rg1 by $\overline{CE}$
SETB P1.4; Rg1 writes #A15-A8=#111b
CLR P1.4
SETB P1.7 ; lock Rg1
CYCPT: MOV R7,#1; counter of 30 parts in buffer
MOV P2,#00000111b; Assign #SB=#1 to read header
CLR P1.6; enable Rg2 by $\overline{CE}$
SETB P1.4; Rg1 writes #SB=A15-A8=#1
CLR P1.4;
SETB P1.6 ; lock Rg2
NXTPRT1:MOV R5,#0; counter of #LB
NXTPRT11:MOV P2,R5; set #LB
REWRD1: CLR P1.3; enable chip SRAM by $\overline{CE}$
CLR P1.1; enable data output from SRAM
CHKHI1:MOV R3,P0; read 1st word from SRAM
MOV R0,R3; copy # of 1st word for further summation
SETB P1.1; disable output of SRAM
SETB P1.3; disable chip of SRAM
CHKHI11: CJNE R3, #7, CHKHI2; check if $w^K_{1,1}$=7
MOV R6,#2; determines number of parts in phrase g=2
AJMP CHKPWD;
CHKHI2: CJNE R3, #123, CHKHI3; check if $w^K_{1,1}$=123
MOV R6,#3; number of parts in phrase g=3
AJMP CHKPWD;
CHKHI3: CJNE R3, #209, NXTPRT2; check if =#209
MOV R6,#4; number of parts in phrase g=4
CHKPWD: INC R5; #LB of 2d word - parole
MOV P2, R5; # to read password
 CLR P1.3; enable chip SRAM by $\overline{CE}$
CLR P1.1; enable data output from SRAM by $\overline{OE}$
PWD:MOV R3,P0; output 2d word from SRAM
SETB P1.1; disable output of SRAM
SETB P1.3; disable chip of SRAM
CJNE R3,#131,NXTPRT2; password =131
AJMP CHKNAME1;
NXTPRT2: DEC R5; return to the 1st word
NXTPRT21:MOV A,R5;
ADD A,#7; #LB for the 1st word in next part
MOV R5,A; #LB
MOV R1,A; copy #LB of word for further calc
CJNE R5,#240,NXTPRT11; check if buffer passed
RETI; buffer is fully processed
CHKNAME1: INC R5; #LB+1
INC R5; #LB increased by 2 for Addresser name 1
MOV P2,R5; #LB to read name1
NOP;
 CLR P1.3; enable chip SRAM by $\overline{CE}$
CLR P1.1; enable data output from SRAM
RENAME1:MOV R3,P0; read name1
SETB P1.1; disable data output from SRAM
SETB P1.3; disable chip of SRAM
CJNE R3, #0, DEC2LB; name1 is =#0
INC R1; #LB for Addresser name2
CHKNAME2: MOV P2,R1; read name2
 CLR P1.3; enable chip SRAM by $\overline{CE}$
CLR P1.1; enable data output from SRAM
RENAME2:MOV R3,P0; read name2
SETB P1.1; disable data output from SRAM
SETB P1.3; disable chip of SRAM
CJNE R3, #0, DEC2LB; name2 is =#0
INC R0; #LB for Addresser name 3
CHKNAME3: MOV P2,R3; read name3
```

```
CLR P1.3; enable chip SRAM by $\overline{CE}$
CLR P1.1; enable data output from SRAM
RENAME3: MOV R3,P0; read next word
SETB P1.1; disable data output from SRAM
SETB P1.3; disable chip of SRAM
CJNE R3, #3, DEC2LB; name 3 =#3
AJMP CHKHSUM1
DEC2LB:   DEC R5;
DEC R5;
DEC R5; return #LB to 1st word in phrase
AJMP NXTPRT21
CHKHSUM1:MOV R1,#0; counter of words
DEC R5;
DEC R5;
DEC R5; return #LB to 1st word in phrase
HSUM1: MOV R0, R5; copy #LB of  word
MOV P2,R5; #LB 1st word of header
MOV A,#0; clear A for summation
CLR P1.3; enable chip SRAM by  $\overline{CE}$
CLR P1.1; enable data output from SRAM
MOV R3,P0; read  word  from SRAM
MOV R1,#0; counter of words in phrase
SETB P1.1; disable output of SRAM
SETB P1.3; disable chip of SRAM
HSUM2: ADD A,R3;
INC R1; counter of words in part
INCR5:INC R5; increment #LB +1
CJNE R1,#8,HSUM1; check counter of words
WRHSUM: MOV R4,A; save header check sum
CJNE R5,#240,RETI
WRCNTPT: MOV R1,#240; write check coef. to Reserve 1
ADD R5,#8; # LB for next part
CJNE R5,#240,RETI
RECNTWD: P2,R5; #LB to read 1st content word
CLR P1.3; $\overline{CE}$ enables chip SRAM
CLR P1.1; $\overline{OE}$ enables output of SRAM
CLR P1.0
MOV P0,R3; read sender name3
SETB P1.0; disable write
SETB P1.1; disable chip SRAM
SETB P1.3
…          …
…          RETI
…          …
```

Output:	buffer Reserve1→	R4 contains total check coefficient for the adequate header $\boldsymbol{W_h}$ = $\{w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}\}$ and is ready to use it further

Carried out microassembler modeling, algorithms, and subroutines complement earlier done designs of AGA operators MINIMUM, MAXIMUM, LITERAL, and C language version of AGA function calculation, published earlier in open access papers [52–54]. For brief approximate comparison of time characteristics one can use reference basic data given in [53], where operators need $t_{\text{MIN-MAX}} \approx 4$ µs and $t_{\text{Literal}} \approx 9$ µs; the calculation of AGA function with $K = 256$ truth levels and $n = 12$ variables takes ≈210,000 work cycles and 0,1 s at 24 MHz. Such time response is comparable with human time response in dialog mode. The computing time for other subroutines can be approximately estimated by the comparison of the number of used operators in various programs.

<u>Results of modelling.</u>

- The proposed structure of coded vocabularies and the conjugated dialog protocol are tested by two microassembler programs, designed for the dual–chip module based on 8-bit microcontrollers MCS-51. They demonstrate principal compatibility of proposed tools with devices of IoT level, provide close complexity level and can be combined together with earlier published algorithms [53,54,59] for the calculations of AGA functions and the base LL scheme.
- The designed dialog phrase protocol for agents communication, involving 8-bytes header and content parts, is compatible and convenient enough for the procession by 8-bit dual-chip module.

- Designed vocabularies structure can be used in simple 8-bit platforms with limited free memory resources, and the proposed dialog protocol can selectively transfer various combinations of logic words and verification coefficients.
- AGA vocabularies structure in 8-bit modelling provides 256 coded words $w_{v,m}^{K}$ and $w_{v,m}^{N}$, what gives the possibility to form complicated enough terminology chains and to design tasks descriptions for distant exchange of data.
- Adapted version of LL with 5 input variables is the minimalistic possible tool for protected local storages of critical data, protected by paroles and hash values. If necessary, intermediate versions of LL can be extended by means of the enlarged number of hash values, approving the LL by other internal and external modules.
- Proposed subroutine LLWRD is compatible with algorithms published earlier in [52,53] and complements AGA subroutines for XOR calculations in [53]. Also one can use the software from for connection to PC.
- Designed algorithms are applicable both for truth levels codes $w_{v,m}^{K}$ and natural numbers ones $w_{v,m}^{N}$, what can be further used for new and more exhausted verification schemes in the heterogeneous logic architecture of agent.

Further research needs to design universal algorithm and microassembler subroutine of template AGA function, which can be flexibly adapted for the procession of all proposed versions of LL scheme, i.e. can be re-switched by the choice of appropriate control parameters. That will help to avoid excessive software in dual-chip module, which is limited by 500-600 operators per one MCS-51. SRAM microscheme in dual-chip module potentially can be enlarged up to 32 MB.

Vocabularies structure and words codes should be further applied to the dual-chip scheme itself and to commercial robotic platforms like Arduino, in order to imitate peripheral AI procedures in decision making module and to involve standard wireless network modules.

7. Discussion

Further research of the proposed vocabularies structure and dialog mode of verification refers to the design of control models for large scale robotic systems, where only administrators and supervising agents possess holistic formal logic model, and separate groups of robots has only fragmentary knowledge and sets of subroutines. In such a system, full-scale modelling in every agent is too expensive and excessive, and the reasonable strategy is to reconfigure the individual agent model for the current task and situation. Then autonomous schemes of missing knowledge search, exchange and verification of reconfigured models become the key element. Another aspect of the problem is to adapt the structure of vocabularies and WM to specialized fragments, appropriate for the design of transformable and self-learning robots with changeable tools. The expected complication here is that verification procedures should be enlarged to monitor the compatibility of robotic hardware platforms, accessible tools, and corresponding software.

Close actual task is the extension of the proposed vocabularies structure to the control of the WM, describing the operation of agent in the scene of action and activating corresponding subroutines and actuators. This goal follows from the review of robotic WMs [62], where the main contribution was to design explicit dimensions of WMs, to determine the underlying principles to make decisions, and to search the trade-offs model for conflict situations. The conclusion of the authors of was that the universal WM does not exist, as it highly depends on the robotic system, the environment in which it operates, and the performed tasks. Even simple robot can use more than one WM, e.g., one for navigation and another one for manipulation. Respectively, the procession of complicated vocabularies structure for $\boldsymbol{V}_{v,m}^{L,N,K}$ is the realistic scheme for WM modeling and can be extended to the selection of activated expressions and software subroutines. However, the choice of vocabularies and words may itself include special calculations and estimations based on expert knowledge. Then WM can be considered as the comprehensive current structure of vocabularies subsets, whose choice is to be done by the decision-maker for the specific current task, basing on rules of work, objects detected in the scene of action, current goals, instructions received from the owner of robots, and behavior of people and robots in the scene. Such research also seems to include complicated AI optimization and AGA metrics, estimating the distance to the formal aim in the multi-parametric space.

Proposed FMF remains here a service tool, which good correlates with the idea of distributed small-scale local memory storages used in the network MAS. The motivated structure of hidden and protected by LL data storages for irregular checks can be preliminary prepared and activated by microcontrollers without special high-level robotic languages. The proposed dialog protocol provides access to data basing on content control, and real addresses for such data storages will be hidden for external agents. Minimalistic versions of LL storages with enough number of passwords should be complemented by confidential digital map of addresses for specific vocabularies sets.

The proposed simple adaptation of basic LL scheme for small scale robotic collective creates the task to select several most actual template versions of LL for enhanced AI modelling, including content analysis and parametrical passwords, adaptable to various external conditions. In small-scale groupes of agents the adaptive structure of LLs is to be attributed to the history of the completed work, can complement one-time quasi random passwords in verification procedures.

As the microcontroller-based communications module and its re-coding tables for $w_{v,m}^{K}$ and $w_{v,m}^{N}$ are divorced from the decision-making module and mass Internet tools for data procession, then one receives the opportunity to cut off incorrect external codes from AGA contours. The development of allocated communication module with extended verification procedures gives the potential possibility to control and to exclude the illegal loading of external codes into the decision-making subsystem of agent via the interface with PC and the network card. Then the design of AGA-controlled verification scheme is to be extended from the communication module also to the interface between the decision-maker and standard USB module with wireless network hardware.

One more actual problem is the slow enough progress in robotic computer vision [13,14] and the real success in fake images generation [83], what limits the development of verification methods. In such extensively developing fields as IoV and the positioning of delivery drones, advanced verification procedure should integrate not only space and time parameters of routes, but also to use formal monitoring of images of cargo, terminals, territorial restrictions, and the biometrics of personnel. Further research is to disclose, how useful can be the conjugation of AGA models and NN. At least it is to compensate the known problem of deliberate data correction in NN [22].

In a small-scale MAS without distinctly pronounced statistical characteristics of agents behaviour, the application of quantum computing for verification seems not to be a priority task. However, this situation can be changed greatly for the global robotic system or for the imitation of human creative algorithms, where it will be necessary to generate and compare many alternative AGA expressions. Then simple in essence AGA vocabularies structure logic modelling and verification algorithms can help to unite quantum procedures and collective robotic tasks.

8. Conclusions

Technical faults, attacks and illegal data modification make distant data verification the obligatory component of collective robotic systems. Verification is considered as the set of data integrity and authentication tests, activated for initial debugging of the agent, periodical testing, work correction, and system restoration. Verification procedures can be initiated not only by a distant administrator or partner robots from the same MAS, but also by state regulator`s agents and loyal or conflicting external robots. Verification tasks motivate to use non-exhausted selective check procedures and distributed local data storages for critical data documenting, protected by LL schemes. Verification should combine automatic and human checks.

In the presented paper robotic communications and verification procedures are firstly proposed to be based on AGA logic model, using ordered high capacity vocabularies structure and formal logic dialog language. AGA-based robotic language and dialog phrases protocol are defined by the ordered matrix structure of vocabularies $V_{v,m}^{N}$ and $V_{v,m}^{K}$ formed by logic words $w_{v,m}$. Firstly, the scene of actions, robots, objects, tasks, tools, and involved software products are to be described by the preliminary formed structure of natural language words and their collocations, represented by vocabularies V^{L} given in the natural language. Further vocabularies V^{L} are to be mapped onto the ones V^N, represented by natural numbers codes, which in its turn should be mapped onto vocabularies V^K formed by truth levels codes defined in AGA. Such three-levels structure of equivalent coding provides the optional use of not protected natural numbers codes and of quasi-random truth levels codes, preventing from illegal data modification, and providing enhanced data integrity checks and the search of errors.

Phrases in dialogs of robots are to be coded by truth levels codes and transmitted by parts, formed by 8-byte sequences of coded numbers. Proposed protocol provides the possibility to find correct content parts of the phrase, what gives the principal possibility to transfer them in parallel even with time delays, e.g., by different laser wavelengths or various radio frequencies in wireless data lines. The procession of the phrase is proposed to be done by AGA-based functions, realizing high capacity finite automatons for the verification of phrases. AGA template matching or logic filtering of input logic phrases is to be used further for the formal preliminary interpretation of phrases. Such procession procedures don`t disclose real physical values of coded internal parameters of agent.

In order to provide reliable documenting of data in the distributed set of local data storages in network agents, the presented paper proposes adapted versions of AGA-based LL. The firstly proposed scheme of mapping of multi-parametric logic AGA functions onto truth levels scale provides small dimensional (1D, 2D, 3D) fragmented projections (or logic diagrams) convenient both for human and automatic checks, as well as for critical data saving in local storages in network agents. Such diagrams can help to visualize and to find errors in verified data.

Wide bands of time, space and some other parameters can be obtained by means of correlated input logic variables taken in AGA, what provides the chance to extend the proposed communications protocol for large-scale systems.

Algorithms and software modelling are demonstrated for the allocated 8-bit microcontroller module, included into the dual-chip microcontroller scheme, earlier used for MVL-LL design. The cascaded scheme of two dual-chip nodules is considered as a platform for communication module for further verification and AI modelling.

Designed MVL-schemes are based on the heterogeneous logic architecture of agent, what gives the methodology to combine AGA, traditional Boolean and fuzzy logic.

Funding: This research received no external funding.

Conflicts of Interest: Author declares no conflict of interest.

Appendix A

Algorithm 1. Phrase procession for dialogs Administrator-Agent and Agent –Agent from the same MAS. Total check coefficients $C^{K/N}$ *or* C^{KXR} are calculated by arithmetic summation or binary XOR operations done for mapped truth levels. Phrase can initiate new dialog or continue the opened one.

Input:	$\boldsymbol{K} \leftarrow$	Number of truth levels in AGA;
	$V_{v,m}^K \leftarrow$	Vocabularies in truth levels representation;
	$(w_{1,4}^K, w_{2,4}^K, w_{3,4}^K) \leftarrow$	Adresser code, given by the triple of coded truth levels in chain vocabularies;
	$(w_{1,5}^K, w_{2,5}^K, w_{3,5}^K) \leftarrow$	Sender codes triple;
	$(p, q, w_{p,q}^K), \dots, (r, s, w_{r,s}^K) \leftarrow$	Set of triples, representing content data given by Decision maker for transfer to Adresser; $p, \dots, q$ and $r, \dots, s$ – arbitrarily given enlarging values, p≤r, q≤s ;
	$w_{1,1}^K \leftarrow$	Welcome code {"Hi"} to begin new session;
	$w_{1,2}^K \leftarrow$	Fixed parole for initial access to Adresser;
	$w_{1,3}^K \leftarrow$	Assigned hash value;
	$L_{al} \leftarrow$	List of allowed contacters
	$L_s \leftarrow$	List of addressers with opened sessions;
	$L_{hash} \leftarrow$	List of accumulated quasi random keys h_{ij} $\in L_{HASH}$ taken from QRNG;

N	Subject	Operation	Commentary
1	Sender $(w^K_{1,5}, w^K_{2,5}, w^K_{3,5})$;	Checks ($w^K_{1,4}, w^K_{2,4}, w^K_{3,4}$) $\in L_{al}$, if yes go to step 2 ; other go to procedure for unknown contacters	*; if list L_{al} of allowed contacts contains Adresser`s name triple:*
2		Assigns i=1, for empty Phrase header template $\boldsymbol{W}_{\text{i,j}}$ ={ … , … , … , … , … , … , … , … };	*; begin new phrase and header*
3		Assigns j=1	*; number of content part*
4		Assigns Format $w^K_{1,1}$ = "Hi" to Phrase header part $\boldsymbol{W}_{\text{i,j}}$ ={ $w^K_{1,1}$, … , … , … , … , … , … , … };	*; initiation of dialog*
5		Assigns Fixed parole $w^K_{1,2}$ of Addresser $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4})$ to phrase header $\boldsymbol{W}_{\text{i,j}}$ ={$w^K_{1,1}, w^K_{1,2}$, , … , … , … , … };	*; takes fixed $w^K_{1,2}$ from $V^K_{1,2}$*
6		Assigns Adresser`s triple name $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4})$ to the header $\boldsymbol{W}_{\text{i,j}}$ ={ $w^K_{1,1}, w^K_{1,2}, w^K_{1,3}, w^K_{2,3}, w^K_{3,3}$, … , … , … };	*; insert triple $V^K_{1,3}, V^K_{2,3}, V^K_{3,3}$*
7		Assigns own Sender`s triple name $(w^K_{1,5}, w^K_{2,5}, w^K_{3,5})$ to the header $\boldsymbol{W}_{\text{i,j}}$ ={ $w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}$};	*; insert triple $V^K_{1,4}, V^K_{2,4}, V^K_{3,4}$*
8		Maps truth levels onto natural numbers scale and calculates total check coefficient $C^{K/N}_{i.j}$ for the header $C^{K/N}_{1,1} = \sum^2_{m=1} w^K_{1,m} + \sum^3_{v=1} \sum^4_{m=3} w^{K/N}_{v,m}$;	
9		Calculates number of content parts: $N_C \geq (r\ x\ s - p\ x\ q)/6$; $N_C \in N$, p, q, r, s are taken from the received content data sequence $\{w^K_{p,q}, \dots, w^K_{r,s}\}$;	*; number of content parts is determined by 6 free fields in any of them*
10		Assigns v=p, m=q	*; set counters of chain vocabularies*
11		Assigns $j = N_C$	*; begin content parts and set their counter*
12		Assigns g=6	*; counter of free fields in a content part*
13		Writes next hash $w^K_{1,3}$ from the list L_{hash} to the first field of phrase content part $\boldsymbol{W}_{\text{i,j}} = \{w^K_{1,3}$, … , … , … , … , … , … , … , … $\}$;	*; begin new content part*
14		Goes to step 18	*; next content part*
15		Checks if $m = s$, *if* yes , goes to step 23 other, goes to step 18	
16		Checks if $v = r$, *if* yes, goes to step 26 other, goes to step 18	*;end of phrase formation*
17		Checks if $j = 0$, *if* yes, goes to step 12 other, goes to step 25	

18		Write $w^K_{p,q}$ into the next vacancy in the content part $\boldsymbol{W}_{\text{i,j+1}} = \{w^K_{1,3}, w^K_{p,q}, w^K_{p+1,q}, w^K_{p+2,q}, \dots, \dots, \dots, \dots, \dots\}$	
19		g=g-1	*; check for free fields*
20		Checks if $g = 0$, if yes, goes to step 25 other, goes to step 21	; $m+1$
21		m = m + 1	
22		Goes to step 15	
23		v=v+1	*; process next v*
24		Goes to step 16	
25		Maps truth levels onto the natural numbers scale and calculates total coefficient $C^K_{i.2}$ of the content part $C^{K/N}_{i.j} = \left(h^{ad}_{ij} + \sum_{v=p}^{q} \sum_{m=r}^{s} w^{K/N}_{v,m}\right)_{content\ part}$	
26		Calculates total summation $C^{K/N}_{total} = C^K_{1,1} + \sum_{i=2}^{Nc} \sum_{j=1}^{Nc} C^{K/N}_{i,j}$	
27		Writes $C^{K/N}_{total}$ into the current content part of phrase $\boldsymbol{W}_{\text{i,j}} = \{w^K_{1,3}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, C^{K/N}_{total}\}$	
28		Sends the header $\boldsymbol{W}_{\text{i,j}}$ and content parts $\boldsymbol{W}_{\text{i,j+1}}, \boldsymbol{W}_{\text{i,j+2}}, \dots$ to Adresser	
29		i=i+1	*; counter of phrases for continued dialog*
30		Waits reply from addresser	
31	Adresser $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4})$	Writes message parts in the buffer: $\boldsymbol{W}_{\text{i,j}} = \{w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}\}$; $\boldsymbol{W}_{\text{i,j+1}} = \{w^K_{1,3}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, w^K_{v,m}, C^{K/N}_{total}\}$ …	*; indexes v, m indicate arbitrarily given words*
32		Reads word $w^K_{1,1}$ in the Format field of the phrase header $\boldsymbol{W}_{\text{i,j}} = \{w^K_{1,1}, w^K_{1,2}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}\}$;	
33		Assigns the number of content part $j = N_C$	
34		Checks, if list L_s of opened sessions contains Adresser`s name $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4})$, declared in the header $\boldsymbol{W}_{\text{i,j}}$: Checks, if $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4}) \in L_s$, if yes, goes to step 38, other, goes to step 35	
35		Checks, if list L_{al} of allowed contacters contains Adresser`s name $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4})$: Checks, if $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4}) \in L_{al}$, if yes, goes to step 37, other, goes to step 36	
36		Send denial of service message	
37		Maps truth levels onto natural numbers scale and calculate total coefficient for the header $C^{*K}_{i,1} = \sum_{m=1}^{2} w^K_{1,m} + \sum_{v=1}^{3} \sum_{m=3}^{4} w^K_{v,m}$;	*; check received external coefficients*
38		Maps truth levels onto natural numbers scale and calculate total coefficient for content parts	

	$C^{*K/N}_{total} = C^{*K}_{i,1} + \sum_{i}^{Nc}\sum_{j=1}^{Nc} C^{*K/N}_{i,j}$	
39	Compares the received and the declared values : Checks, if $C^{*K}_{total} = C^{K}_{total}$, if yes , goes to step 40, other, goes to step 36	
40	Sends received phrase content $(p, q, w^K_{p,q}), \ldots, (r, s, w^K_{r,s})$ to Decision maker	*; excluding hashes and check coefficients*
41	Waits reply from Decision maker	
42	Receives reply $(p, q, w^K_{p,q}), \ldots, (r, s, w^K_{r,s})$ from Decision maker	*;for simplicity indexes are the same as in step 32*
43	Assigns next quasi random hash from the list L_{hash} to $w^K_{1,2} = h_{ij}$	
44	Assigns i=1 for empty phrase template $\boldsymbol{W}_{i,j}$ ={ $\ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots$ }	*; form reply phrase*
45	Assigns j=1	
46	Assigns Format $w^K_{1,1}$ = "Received" to phrase header $\boldsymbol{W}_{i,j}$ ={ $w^K_{1,1}, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots$ }	*; format differs from request*
47	Assigns Fixed parole $w^K_{1,2}$ of Sender to phrase header $\boldsymbol{W}_{i,j}$ ={$w^K_{1,1}, w^K_{1,2}$, , $\ldots, \ldots, \ldots, \ldots$ }	*; takes fixed $w^K_{1,2}$ from $V^K_{1,2}$*
48	Assigns Sender`s triple $(w^K_{1,5}, w^K_{2,5}, w^K_{3,5})$ to Phrase header part $\boldsymbol{W}_{i,j}$ ={ $w^K_{1,1}, w^K_{1,2}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}, \ldots, \ldots, \ldots$ }	
49	Assigns own code $(w^K_{1,4}, w^K_{2,4}, w^K_{3,4})$ to Phrase header part $\boldsymbol{W}_{i,j}$ ={ $w^K_{1,1}, h^{ad}_{ij}, w^K_{1,5}, w^K_{2,5}, w^K_{3,5}, w^K_{1,4}, w^K_{2,4}, w^K_{3,4}$};	
50	Assigns j=j+1	
51	Maps truth levels onto natural numbers scale and calculate total check coefficient $C^{K/N}_{i.j}$ for the header $C^{K/N}_{1,1} = \sum_{m=1}^{2} w^K_{1,m} + \sum_{v=1}^{3}\sum_{m=3}^{4} w^{K/N}_{v,m}$;	
52	Calculates number of content parts: $N_C \geq (r\ x\ s - p\ x\ q)/6$; $N_C \in N$, p, q, r, s are taken from the received content data sequence $\{w^K_{p,q}, \ldots, w^K_{r,s}\}$;	*; number of content parts is determined by 6 free fields in a content part*
53	Assigns v=p, m=q	*; set counters of chain vocabularies*
54	Assigns $j = N_C$	*; begin content parts and set their counter*
55	Assigns g=6	*;counter of free fields in a content part*
56	Writes next Hash $w^K_{1,3}$ from the list L_{hash} to the first field of phrase content part $\boldsymbol{W}_{i,j+1} = \{w^K_{1,3}, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots\}$;	*;begin new content part*
57	Goes to step 61	*;begin next content part*

58	Checks, if $m = s$, *if* yes, goes to step 66, other, goes to step 61	*; check counter*
59	Checks, if $v = r$, *if* yes, goes to ste other, goes to step 61	*;end of phrase formation*
60	Checks, if $j = 0$, *if* yes, goes to step 55 other, goes to step 68	*; check counter*
61	Writes set of $w_{v,m}^{K}$ into the next vacancy in the content part $\boldsymbol{W}_{\mathrm{i,j}} = \{w_{1,3}^{K}, w_{v,m}^{K}, \ldots, \ldots, \ldots, \ldots, \ldots\}$	
62	Assigns g=g-1	*; free fields control*
63	Checks, if $g = 0$, *if* yes, goes to step 68, other, goes to step 64	$;m+1$
64	Assigns $\mathrm{m} = \mathrm{m} + 1$	
65	Goes to step 58	
66	Assigns v=v+1	*; process next v*
67	Goes to step 59	
68	Maps truth levels onto natural numbers scale and calculate total coefficient $C_{i.2}^{K}$ of content part $C_{i.j}^{K/N} = \left(h_{ij}^{ad} + \sum_{v=p}^{q} \sum_{m=r}^{s} w_{v,m}^{K/N}\right)_{content\ part}$	
69	Calculate total summation $C_{total}^{K/N} = C_{1,1}^{K} + \sum_{i}^{Nc} \sum_{j=1}^{Nc} C_{i,j}^{K/N}$	
70	Writes C_{total}^{K} into the current content part of phrase $\boldsymbol{W}_{\mathrm{i,j}} = \{w_{1,3}^{K}, w_{v,m}^{K}, w_{v,m}^{K}, w_{v,m}^{K}, w_{v,m}^{K}, w_{v,m}^{K}, w_{v,m}^{K}, C_{total}^{K/N}\}$	
71	Replies to Sender and transfer the header $\boldsymbol{W}_{\mathrm{i,j}}$ and content part $\boldsymbol{W}_{\mathrm{i,j+1}}$	
72	i=i+1	
73	Waits for the next dialog phrase	
Output: →	Formation and transfer of the request $\{\boldsymbol{W}_{1,1}, \boldsymbol{W}_{1,2}\}$ to Adresser, and procession of reply $\{\boldsymbol{W}_{2,1}, \boldsymbol{W}_{2,2}\}$ to Sender, saving total check coefficients $C_{i,j}^{K/N}$ in memory for the continuation of the dialog.	

References

1. El Madani, S., Motahhir, S.; El Ghzizal, A. Internet of vehicles: concept, process, security aspects and solutions. *Multimed Tools Appl.* **2022**, *81*, 16563–16587. https://doi.org/10.1007/s11042-022-12386-1
2. Daki, H.; El Hannani, A.; Aqqal, A.; Abdelfattah Haidine, A.; Dahbi, A. Big Data management in smart grid: concepts, requirements and implementation. *J Big Data* **2017**, 4, 13, 19 pages. https://doi.org/10.1186/s40537-017-0070-y
3. Patrao, C.; Moura, P.S.; . De Almeida, A.T. Review of Smart City Assessment Tools. **2020**, *Smart Cities*, 3, 4, 1117-1132. DOI: 10.3390/smartcities304005
4. Zhang,K.; Chermprayong,P.; Xiao, F.; Tzoumanikas, D.; Dams, B.; Kay, S.; Kocer, B.B.; Burns, A.; Orr, L.;Alhinai,T.; Choi, C.; Darekar, D. D.; Li, W.; Hirschmann, S.; Soana, V.; Ngah, S.A.; Grillot, C.; Sareh, S.; Choubey, A.; Margheri, L.; Pawar, V. M.; . Ball, R.J.; Williams, C.; Shepherd, P.; Leutenegger, S.; Stuart-Smith R.; Kovac M. Aerial additive manufacturing with multiple autonomous robots. *Nature* **2022**, *609*, 709–717. https://doi.org/10.1038/s41586-022-04988-4
5. Khanna, A., Kaur, S. Internet of Things (IoT), Applications and Challenges: A Comprehensive Review. *Wireless Pers Commun* **2020**,*114*, 1687–1762 . https://doi.org/10.1007/s11277-020-07446-4
6. Maalouf,N.; Sidaoui,A.; Elhajj, I.H.; Asmar, D. Robotics in Nursing: A Scoping Review, *Journal of Nursing Scholarship* **2018,** *50*, 6, 590-600. DOI: 10.1111/jnu.12424

7. Luckcuck,M.; Farrell, M.; Dennis, L. A.; Dixon, C.; Fisher, M. Formal Specification and Verification of Autonomous Robotic Systems: A Survey. *ACM Comput. Surv*. **2019**,52, 5, Article 100, 41 pages. https://doi.org/10.1145/3342355
8. Araujo,H.; Mousavi M. R.; Varshosaz, M. 2023. Testing, Validation, and Verification of Robotic and Autonomous Systems: A Systematic Review . *ACM Trans. on Software Eng. and Methodol*. **2023**, *32*, 2, Article 51, 1–61. https://doi.org/10.1145/3542945
9. Kok Yeow You. A Summary on 5G and Future 6G Internet of ThingsMarch 2023 In Book: *Opportunities and Challenges of Industrial IoT in 5G and 6G Networks*, Ed. by Yu, P., Hu, X., Prakash, A., Misuko, N. W., Haiyue, G.; Ch. 10, 196-243. Hershey, PA: IGI Global, 2023. DOI: 10.4018/978-1-7998-9266-3.ch010
10. Nikolic, G.; Dimitrijević, B.; Nikolic, T.; Stojcev, M.Fifty years of microprocessor evolution: from single CPU to multicore and manycore systems *FACTA UNIVERSITATIS Ser. Electronics and Energetics* ***2021***, *35*, 2,155-186. DOI: 10.2298/FUEE2202155N.
11. *Shreyas D. K., Srivatsa N. Joshi, Vishwas H. Kumar, Vishaka Venkataramanan, Kaliprasad C. S*Joshi, S.N.; Kumar,V.H.; Venkataramanan, V.; Kaliprasad C S A Review on Neural Networks and its Applications September 2023 *J. of Comp. Technol. & Applic.* **2023**, *14*, 2. DOI: 10.37591/jocta.v14i2.1062
12. Strzelecki,R.M.; Demidova, G.; Lukichev,D.; Poliakov, N.; Abdullin, A.; Lovlin, S. Survey on fuzzy logic methods in control systems of electromechanical plants. January 2019 *Scientif. and Tech. J. of Inf. Techn. Mech. and Optics* **2019**, *19*, 1,1-14. DOI: 10.17586/2226-1494-2019-19-1-1-14
13. Shreya Shelke, Indrayani S. Pathak, Aniket P. Sangai, Dipali Lunge, Kalyani Shahale, Harsha R. Vyawahare. A Review Paper on Computer Vision. *Int. J. of Adv. Res. in Science Commun. and Techn.* **2023**, *3*, 2, 673-676. DOI: 10.48175/IJARSCT-8901
14. Elyan, E: Vuttipittayamongkol, P.; Johnston, P.; Martin, K.; McPherson, K.; Moreno-García, C.F.; Jayne, C.; Md. Mostafa Kamal Sarker. Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward. *Art. Int. Surg*. **2022**, 2, 24-45. http://dx.doi.org/10.20517/ais.2021.15
15. Basak,S.; Agrawal,H.; Jena, S.; Gite, S.; Bachute, M.; Pradhan, B., Assiri, M.Challenges and Limitations in Speech Recognition Technology: A Critical Review of Speech Signal Processing Algorithms, Tools and Systems. *Comp. Modeling in Eng. & Sci.* **2022**, *135*, 2, 1-37. DOI: 10.32604/cmes.2022.021755
16. García, S.; Ramírez-Gallego, S.; Luengo, J; Benítez J.M.; Herrera, F. Big data preprocessing: methods and prospects. *Big Data Anal.* **2016**, *1*, 9, 22 pages. https://doi.org/10.1186/s41044-016-0014-0
17. Jency Rubia J, Babitha Lincy R, Ezhil E Nithila, Sherin Shibi, Rosi A. A Survey about Post Quantum Cryptography Methods. *EAI Endorsed Trans. on Internet of Things*, **2024**,*10*, 9 pages,. DOI: 10.4108/eetiot.5099
18. Bykovsky, A.Y.; Kompanets, I.N. Quantum cryptography and combined schemes of quantum cryptography communication networks. *Quantum Electron.* **2018**, *48*, 777–801, https://doi.org/10.1070/qel16732.
19. H Shah, Rashid Ali Laghari, Kamlesh Kumar, A A Waqan, Awais Khan Jumani. A Review on Quantum Computing Trends & Future Perspectives *EAI Endorsed Trans. on Cloud Systems* **2022**, *7* , 22, 11 pages. DOI: 10.4108/eai.17-5-2022.173979
20. Thiruthanigesan Kanagasabai, Nagarathnam Thiruchchelvan. Data Verification and Validation Process in the Management System Development. *Middle East J. of Scientific Res.* **2017**, *25*, 5, 902-911.DOI: 10.5829/idosi.mejsr.2017.902.911
21. Ricardo Caldas ; Juan Antonio Piñera García, Matei Schiopu, Patrizio Pelliccione, Genaína Rodrigues, Thorsten Berger. Runtime Verification and Field Testing for ROS-Based Robotic Systems. arXiv:2404.11498v1 [cs.SE] 17 Apr 2024.
22. Bykovsky, A.Y. Multiple-Valued Logic and Neural Network in the Position-Based Cryptography Scheme. *J.of Russian Laser Res.* **2021**, *42*, 618–630. https://doi.org/10.1007/s10946-021-10000-7
23. Chen, J.; Micali, S. Algorand: A Secure And Efficient Distributed Ledger. *Theoretical Computer Science*, **2019**, *777*, 155–183. https://doi.org/10.1016/j.tcs.2019.02.001
24. Ethan,B.; Jae, K. Cosmos Whitepaper: A Network Of Distributed Ledgers. https://v1.cosmos.network/resources/whitepaper, 2016.
25. C. Cardoso,R. C.; Farrell, M.; Luckcuck, M.; Ferrando, A.; Fisher. M. Heterogeneous Verification of an Autonomous Curiosity Rover. arXiv.2007.10045v1 [cs.SE] 20 Jul 2020. (accessed 12 November 2024)
26. Aminof, B., Murano, A., Rubin, S., Zuleger, F. (2015). Verification of Asynchronous Mobile-Robots in Partially-Known Environments. Eds. Chen, Q., Torroni, P., Villata, S., Hsu, J., Omicini, A. PRIMA 2015: Principles and Practice of Multi-Agent Systems. PRIMA 2015. Lecture Notes in Computer Science, volume 9387. Springer, Cham. 185–200. 2015.https://doi.org/10.1007/978-3-319-25524-8_12
27. Kouvaros, P.; Lomuscio, A. Formal verification of opinion formation in swarms. In Proc. of the 15 International Conference on Autonomous Agents & Multiagent Systems (AAMAS 2016). Singapore, 9-13 May, 2016,1200–1208.
28. Foughali, M.; Berthomieu, B.; Dal Zilio, S.; Hladik, P.-E.; Ingrand, F.; Mallet, A.. Formal verification of complex robotic systems on resource-constrained platforms. In 2018 IEEE/ACM 6th International FME

Workshop on Formal Methods in Software Engineering (FormaliSE). IEEE, Gothenburg, Sweden, May 27–3 June 2018. pp. 2–9. https://doi.org/10.1145/3193992.3193996
29. Kamali M.; Dennis, L. A.; McAree, O.; Fisher, M.;. Veres, S. M. 2017. Formal verification of autonomous vehicle platooning. *Science of comp. programming* **2017**,*148*, 88–106.
30. Barbot, B.; Bérard, B.; Duplouy, Y.; Haddad, S.. Statistical model-checking for autonomous vehicle safety validation. In Conference SIA Simulation Numérique. HAL-Inria, Montigny-le-Bretonneux, France. 16 March 2017 doi:hal-01491064.
31. Luo C.; Wang, R.; Jiang, Y.; Yang, K.; Guan, Y.; Li, X.; Shi. Z. 2018. Runtime verification of robots collision avoidance case study. In IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Vol. 1. IEEE, Tokyo, Japan, 23- 27 July 2018, pp.204–212. DOI:10.1109/COMPSAC.2018.00033
32. Lomuscio, A.; Michaliszyn, J. 2015. Verifying Multi-Agent Systems by Model Checking Three-valued Abstractions. In Proc. of the 14th Int. Conf. on Autonom. Agents and Multiagent Systems (AAMAS 2015), Eds. Bordini, E.;, Weiss, Y; May 4–8, 2015, Istanbul, Turkey, vol. 15, 189–198.
33. Mansoor,N.; Saddler, J. A.; Silva, B.; Bagheri, H.; Cohen, M. B.; Farritor, S.. Modeling and testing a family of surgical robots: an experience report. In Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, Lake Buena Vista, FL, USA, November 4 - 9, 2018 , 785–790.
34. Cavalcanti, A.; Miyazawa, A.; Sampaio, A.; Li, W.; Ribeiro, P.; Timmis, J. Modelling and verification for swarm robotics. In Proc. Int. Conf. on Integr. Formal Methods. Springer, Maynooth, Ireland, September5-7, 2018,1–19. 2018
35. Betts, K. M.; Petty, M. D. 2016. Automated search-based robustness testing for autonomous vehicle software. *Modelling and Simulation in Engineering* **2016** , 3, 1-15.DOI:10.1155/2016/5309348
36. Brambilla, M.; Brutschy, A.; Dorigo, M.; Birattari, M. Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems* (TAAS) **2014**,*9*, 4 , 1–28. DOI:10.1145/2700318
37. O'Kelly, M.; Abbas, H.; Gao, S.; Shiraishi, S.; Kato, S.; Mangharam, R.. APEX: Autonomous vehicle plan verification and execution. In Proc. SAE 2016 World Congress and Exhibition. SAE International, April 12-14 2016, Detroit,MI, USA,vol. 1, pp. 1–13.
38. Desai, A.; Dreossi, T.; Seshia, S. A. 2017. Combining Model Checking and Runtime Verification for Safe Robotics. In Runtime Verification, Eds. Lahiri S., Reger G. Springer Internat. Publishing, Cham, 2017, 172–189.
39. Lu, Y.; Niu, H.; Savvaris, A.; Tsourdos,A. 2016. Verifying collision avoidance behaviours for unmanned surface vehicles using probabilistic model checking. IFAC-PapersOnLine **2016,** *49*, 23, 127–132.
40. Foughali, M.; Berthomieu, B.; Dal Zilio, S.; Ingrand, F.; Mallet, A. Model checking real-time properties on the functional layer of autonomous robots. In Int. Conf. on Formal Engineering Methods ICFEM 2016. Springer, Tokyo, Japan, November 14-18, 2016, 383–399.
41. Hagerman, S.; Andrews, A.; Oakes, S. Security testing of an unmanned aerial vehicle (UAV). In Proc.2016 Cybersecurity Symposium (CYBERSEC). IEEE, Coeur d'Alene, ID, USA,18-20 April 2016, 26–31. doi: 10.1109/CYBERSEC.2016.012.
42. Abdelsalam, M.; Khalil, K.; Stickley, J.; Salem, A.; Loye, B. Verification of advanced driver assistance systems (ADAS) and autonomous vehicles with hardware emulation-in-the-loop. Int. J. of automotive eng. **2019**,*10*, 2, 197–204. DOI:10.20485/jsaeijae.10.2_197.
43. Ben Abdessalem, R.; Panichella, A.; Nejati, S.; Briand, L. C. ; Stifter T. 2018. Testing autonomous cars for feature interaction failures using many-objective search. In 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, Montpellier, France, September 3 - 7, 2018,143–154. DOI:10.1145/3238147.3238192
44. Sieber, C. ; Vieira da Silva, L. M.; Grünhagen, K.; Fay, A.. Rule-Based Verification of Autonomous Unmanned Aerial Vehicles. *Drones* **2024**, *8*,1, 26. https://doi.org/10.3390/drones8010026
45. Yaacoub E; Abualsaud, K.; Mahmoudm, M. Hybrid Encryption for Securing and Tracking Goods Delivery by Multipurpose Unmanned Aerial Vehicles in Rural Areas Using Cipher Block Chaining and Physical Layer Security. *Drones* **2024**, *8*, 3, 111. https://doi.org/10.3390/drones8030111
46. Ma, J.; Chen, P.; Xiong, X.; Zhang, L.; Yu, S.; Zhang,D. Research on Vision-Based Servoing and Trajectory Prediction Strategy for Capturing Illegal Drones. *Drones* **2024**, *8*, 4,127. https://doi.org/10.3390/drones8040127
47. Gao, J.; Liu, Q.; Chen, H.; Deng,H.; Zhang,L.; Sun, L.; Huang, J. Digital Battle: A Three-Layer Distributed Simulation Architecture for Heterogeneous Robot System Collaboration. *Drones* **2024**, *8*,4 ,156. https://doi.org/10.3390/drones8040156
48. Shrestha, A.; Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **2019**, Shrestha, A., & Mahmood, A. (). Review of Deep Learning Algorithms and Architectures. *IEEE Access*, **2019,** *7*, 53040–53065. DOI: 10.1109/ACCESS.2019.2912200

49. Sun, T.; Cao, J. Research on Machine Vision System Design Based on Deep Learning Neural Network *Wireless Comm. and Mobile Comp.* **2022**, *4*, 1-16 DOI: 10.1155/2022/4808652
50. Chai, G. ;. Cao, Z.; Liu, W.; Wang, S.;Huang, P.; Zeng, G. Parameter estimation of atmospheric continuous-variable quantum key distribution. *Phys. Rev. A* **2019,** *99*, 3, 032326. DOI: 10.1103/PhysRevA.99.032326
51. Allen, C.M.; Givone, D.D. The Allen-Givone Implementation Oriented Algebra. In *Computer Science and Multiple-Valued Logic: Theory and Applications*; D.C. Rine, Ed.; North Holland: Amsterdam, The Netherlands, 1984, 262–283.
52. Bykovsky, A.Yu. Heterogeneous network architecture for integration of AI and quantum optics by means of multiple-valued logic. *Quantum Rep.* **2020**, *2*, 126–165. http://dx.doi.org/10.3390/quantum2010010
53. Bykovsky, A. Multiple-Valued Logic Modelling for Agents Controlled via Optical Networks. *Appl. Sci.* **2022**, *12*, 3, 1263. https://doi.org/10.3390/app12031263
54. Bykovsky, A.Y.; Vasiliev, N.A. Data Verification in the Agent, Combining Blockchain and Quantum Keys by Means of Multiple-Valued Logic. *Appl. Syst.Innov*. **2023**, *6*, 51. https://doi.org/10.3390/asi6020051
55. Bruns, G., Godefroid, P.: Model Checking Partial State Spaces with 3-Valued Temporal Logics. In Proc. Computer Aided Verification, 11th International Conference, CAV '99, Trento, Italy, July 6-10, 1999, Eds. Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 274–287. Springer, Heidelberg,1999. DOI:10.1007/3-540-48683-6_25
56. Gurfinkel, A.; Chechik, M. Multi-Valued Model Checking via Classical Model. *Lecture Notes in Computer Science* **2003,** 2761, 17 pages. DOI: 10.1007/978-3-540-45187-7_18.
57. Gurfinkel, A., Chechik, M. (2003). Multi-valued Model Checking via Classical Model Checking. In Proc. CONCUR 2003 - Concurrency Theory. CONCUR 2003. Lecture Notes in Computer Science, Eds. Amadio, R., Lugiez, D., volume 2761. pp.266-280. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45187-7_18
58. Zhu, J.; Li, Y. Hesitant Fuzzy Linguistic Aggregation Operators Based on the Hamacher t-norm and t-conorm. *Symmetry* **2018**, *10*, 189. https://doi.org/10.3390/sym10060189.
59. Bykovsky, A.Y.; Vasiliev, N.A. Parametrical T-Gate for Joint Processing of Quantum and Classic Optoelectronic Signals. *J –Multidisc.Sci. J*, **2023**, 6, 384–410. https://doi.org/10.3390/ j6030026
60. Bykovsky, A. A Multiple-Valued Logic for Implementing a Random Oracle and the Position-Based Cryptography. *J. of Rus. Laser Res.* **2019**, *40*,173-183. DOI: 10.1007/S10946-019-09785-5
61. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*; 4th ed.; Hoboken : Pearson, 2021. Chs.2, 7,11,18, 26.
62. Sakagami, R.; Lay, F. S.; Dömel, A.; Schuster, M. J.; Albu-Schäffer, A.; Stulp, F. Robotic world models—conceptualization, review, and engineering best practices. *Front. Robot. AI, Sec. Robotic Control Systems*. **2023**,*10 – 2023*,1253049, 24 pages. https://doi.org/10.3389/frobt.2023.1253049.
63. Casey, A.; Davidson E.; Poon, M.; Dong, H., Duma, D.; Grivas, Grover, A.C. Su´arez-Paniagua, V.; Tobin, R.; Whiteley, W.; Wu, H.; Alex, B. A Systematic Review of Natural Language Processing Applied to Radiology Reports. arXiv:2102.09553v1 [cs.CL] 18 Feb 2021 (accessed on 12 November 2024)
64. Sun, C.; Sun, P. ; Wang, J.; Guo, Y.; Zhao, X. Understanding LiDAR Performance for Autonomous Vehicles Under Snowfall Conditions," *IEEE Trans. on Intel. Transportation Syst.*, **2024**, *25*, 11, pp. 16462-16472. doi: 10.1109/TITS.2024.3409907.
65. Shahraji, M. H.; Larouche, C. Case Study: Rigorous Boresight Alignment of a Marine Mobile LiDAR System Addressing the Specific Demands of Port Infrastructure Monitoring. *Marine Geodesy* **2022**,.1-18, 10.1080/01490419.2022.2025503
66. Prabadevi, B.; Deepa, N.; Pham, Q.-V.; Nguyen, D.C.; Praveen Kumar Reddy, M.; Thippa Reddy, G.; Pathirana, P.N.; Dobre, O. Toward Blockchain for Edge-of-Things: A New Paradigm, Opportunities, and Future Directions. *IEEE Internet Things Mag*. **2021**, 4, 102–108. https://doi.org/10.1109/IOTM.0001.2000191
67. Zhuang, P.; Zamir, T.; Liang, H. Blockchain for Cybersecurity in Smart Grid: A Comprehensive Survey. *IEEE Trans. Ind. Inform.* **2021**, *17*, 1, 3–19. DOI:10.1109/TII.2020.2998479
68. Ankalkoti, P. A Relative Study on Bitcoin Mining. *Imperial J. Interdiscip. Res.* **2017**, *3*, 5, 1757-1761.
69. Ma, X.; Yuan, X.; Cao, Z.; Qi, B.; Zhang, Z. Quantum random number generation. npj Quantum Inf. 2016, 2, 16021. DOI:10.1038/npjqi.2016.21
70. Conti, M.; Dragoni, N.; Lesyk, V. A Survey of Man in the Middle Attacks. *IEEE Communications Surveys and Tutorials* **2016**, *18*, 3, 2027-2051.doi: 10.1109/COMST.2016.2548426
71. Beebe, N.H. The Mathematical-Function Computation Handbook; Springer: Berlin/Heidelberg, Germany, 2017. https://doi.org/10.1007/978-3-319-64110-2
72. Cyranski, J.F. Measurement theory for physics. *Found. Phys.* ***1979***, *9*, 641–671. https://doi.org/10.1007/BF00711102
73. Hetmański M. Expert Knowledge: Its Structure, Functions and Limits. *Studia Humana* **2018,** *7*, 3,11-20. DOI:10.2478/sh-2018-0014

74. Gua, H. A Mathematical Theory of Language. *Int. J. of Contemporary Educat.* **2017**, *1*, 1-11. DOI: 10.11114/ijce.v1i1.2893
75. Lund, B. D., Wang, T. Chatting about ChatGPT: How may AI and GPT impactacademia and libraries? *Library Hi Tech News*, **2023**, *40*, 3, 26-29. DOI: 10.1108/LHTN-01-2023-0009
76. Antipov, A.L.; Bykovsky, A.Y.; Vasiliev, N.A.; Egorov, A.A. Multiple-valued logic-protected coding for an optical non-quantum communication line. *J. Russ. Laser Res.* **2006**, *27*, 492–505. DOI:10.1007/s10946-006-0031-y
77. MCS 51 Microcontrollers Family User's Manual. https://web.mit.edu/6.115/www/document/8051.pdf (accessed on 12 November 2024)
78. Gregersen, C. A complete guide to microcontrollers for IoT (July 23 2020). https://www.nabto.com/iot-microcontroller-guide/ (accessed on 10 November 2024)
79. Billman. G.E.; Homeostasis: The Underappreciated and Far Too Often Ignored Central Organizing Principle of Physiology. *Front. Physiol*. Sec. Integrative Physiology, **2020** , *11-2020*, 12 pages. https://doi.org/10.3389/fphys.2020.00200
80. Kumar, C. K. Suyambulingom, C. (2012) "Cryptographic of high Security HashFunctions". Int. J. of Eng. Res. & Techn. (IJERT), **2012**, *1*, 3, 7 pages.https://www.ijert.org/research/cryptographic-of-high-security-hash-functions-IJERTV1IS3074.pdf
81. Weiwei, Z.; Baoqiang, L.; Xiuyi, L. A Star Pattern Recognition Algorithm Based on the Radial Companion-Circumferential Feature, *Mathematical Problems in Engineering*, **2022**, 1857481, 10 pages. https://doi.org/10.1155/2022/1857481
82. Arduino. Availabe online: https://www.arduino.cc (accessed on 12 November 2024).
83. Sha, Z.; Li, Z.; Yu, N.; Zhang, Y. DE-FAKE: Detection and Attribution of Fake Images Generated by Text-to-Image Generation Models. In CCS '23: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security , ACM SIGSAC Conference on Computer and Communications Security, Copenhagen Denmark, November 26 - 30, 2023. pp. 3418 – 3432. https://doi.org/10.1145/3576915.3616588