DOI: 10.17587/prin.13.440-448

Self-Synthesis of Programs Based on Artificial Chemistry Model

E. A. Kol'chugina, kea_sci@list.ru, Department of Mathematical Support and Computer Application, Penza State University, Penza, 440026, Russian Federation

Corresponding author:

Elena A. Kol'chugina, D. Sci., Professor, Department of Mathematical Support and Computer Application, Penza State University, Penza, 440026, Russian Federation
E-mail: kea_sci@list.ru

Received on May 15, 2022
Accepted on September 05, 2022

Fully automatic software synthesis will become possible when simpler programs or program components become able to spontaneously and inevitably attract each other and connect with each other. To achieve this, it is necessary to construct special means to provide spontaneous interactions between programs, since currently there are no such means. In this article, using the principles of artificial chemistry, we propose an algebra of "sinks-and-sources", a concept of artificial atom and a model called H_2O. We also describe experiments with this model, in which the formation of a model molecule of "water" is reproduced. We represent artificial atoms of simple substances, such as oxygen and hydrogen, as independent processes corresponding to two types of programs. We construct connections between individual atoms using sockets, which are necessary to simulate shared particles from outer orbitals. During the experiments, we registered the emergence of artificial molecules of "water" and other complex substances. The experimental results prove that simple software units implementing the proposed principles are capable of spontaneously forming complex software structures without directed external influence. The achieved results are useful to software engineering, artificial life and artificial intelligence to provide self-development of software and complex logical structures capable of further evolution and self-improvement.

Keywords: *automatic software synthesis, spontaneous self-formation of complex programs, artificial chemistry, artificial atom*

For citation:

Kol'chugina E. A. Self-Synthesis of Programs Based on Artificial Chemistry Model, *Programmnaya Ingeneria,* 2022, vol. 13, no. 9, pp. 440—448.

УДК 004.42

Е. А. Кольчугина, д-р техн. наук, доц., проф. кафедры, kea_sci@list.ru,
ФГБОУ ВО "Пензенский государственный университет"

Самосинтез программ на основе модели искусственной химии

Полностью автоматический синтез программного обеспечения станет возможным, когда более простые программы или программные компоненты будут способны спонтанно и неотвратимо притягивать друг друга и соединяться. Для этой цели необходимо сконструировать специальные средства обеспечения спонтанного взаимодействия между программами, поскольку в настоящее время таких средств нет. В статье на основе принципов искусственной химии предложены алгебра "стоков" и "источников", концепция искусственного атома и модель под названием H_2O. Также описаны эксперименты с этой моделью, в которых воспроизводится *образование модельной молекулы "воды". Искусственные атомы простых веществ, таких как кислород и водород, представлены как независимые процессы, соответствующие двум типам программ. Соединения между отдельными атомами строятся с помощью сокетов, которые необходимы для имитации общих частиц с внешних орбиталей. В ходе экспериментов было зарегистрировано появление искусственных молекул "воды" и других сложных веществ. Экспериментальные результаты доказали, что простые программные единицы, реализующие предложенные принципы, способны самопроизвольно формировать сложные программные структуры без направленного внешнего воздействия. Достигнутые результаты полезны для программной инженерии, искусственной жизни и искусственного интеллекта для саморазвития программного обеспечения и сложных логических структур, способных к дальнейшей эволюции и самосовершенствованию.*

Ключевые слова: *автоматический синтез программного обеспечения, спонтанное самообразование сложных программ, искусственная химия, искусственный атом*

Introduction and background

Software is an important component of modern technologies, having a critical impact on their quality and efficiency. Therefore, the enhancement of software, its paradigms and principles of development, in turn, is important. The reality of the modern software development process has two significant aspects.

Firstly, this is an aspect related to the organization and technologies of software development. Modern software is created by a team of developers who have different specializations of the programmer profession and often do not interact directly with each other. Sometimes the complexity and scale of development is such that the project cannot be fully comprehend and controlled by one person. This complicates the software development process and reduces its reliability due to influence of the limitations based on the human factor.

Secondly, there is an aspect concerning the life cycle of software. Programs tend to become obsolete quickly, and during exploitation they also need to be improved and adapted. The amount of work on the revision and adaptation of the software may be insignificant, but these works require the involvement of specialists and time-consuming. This means spending additional resources to create and maintain the project, which reduces its effectiveness.

One of the possible solutions is to create programming paradigms and technologies that allow to automate the development, improvement and generation of new versions. This process should take place under supervision, but without the constant direct participation of a specialist.

In this article, we propose a mechanism that can provide spontaneous self-development of programs under certain conditions. This mechanism is designed to ensure the formation of program structures based on independent units endowed with specific means of interaction.

The scope of application of the proposed mechanism for self-development of programs can be arbitrary, regardless of the application. The proposed mechanism is intended for the development of general-purpose software and is compatible with imperative programming tools, allowing the use of such popular languages as C/C++, Pascal, Perl.

Chemistry serves as a metaphor for creating such a mechanism. Programs should be formed as a result of interactions with other programs, similar to the formation of substances in chemistry as a result of reactions. Among computer models, the closest analogies are models of artificial chemistry.

In artificial chemistry models [1—3], programs are considered as artificial molecules capable to react with each other. As a result of the reactions, the molecules (i. e. programs) can combine to form larger and more complex molecules or programs. So, using the terminology of chemistry, the essence of solving the problem of self-development of programs is to provide conditions for the spontaneous synthesis of complex artificial molecules from simpler ones.

To achieve this, we must possess means to artificially reproduce forces of attraction and repulsion to make self-synthesis of artificial molecules selective and asymmetric. We also need to endow our artificial molecules with energy. The standard analogue of energy in artificial life and artificial chemistry models [2, 3] is central processor time. To get energy, artificial molecules, or programs, must be activated as executing processes. To retain energy, the processes should strive to extend their execution as long as it possible. A successful strategy for obtaining such a result is the formation of cyclic structures, as in living systems.

Life is an existence in a form of dynamically stable cyclic repetition of the same set of physiology processes or, in simpler case, reactions. Such an existence possible as long as energy needed to carry out the processes (reactions) is available. That form of existence observed both in natural biological life [4] and in supposed non-protein forms of living, e. g. based on nuclear reactions occurring inside the stars [5], or on computations in artificial life models [3, 6—8].

In models of artificial life and artificial chemistry, cyclic structures representing sets of coupled reactions or artificial analogues of molecular organisms must earn demanded amount of energy by performing specified calculations. This simultaneously prolongs the existence of structures and makes them capable of evolutionary development.

The model of artificial life proposed in [8] was aimed at studying the emergence of such cyclic computing structures. In the experiments described in [8], it was shown that in distributed systems, cyclic structures can spontaneously arise, which manifest themselves in the form of repetitive executions of indirectly interacting processes (functions).

These structures arising in presence of shared variables, which are necessary as input data for processes. But the arising structures are fragile. The cycles are easily corrupted or completely destroyed by competing processes that avariciously searching for the same input data.

The problem outlined in [3, 8] is that there are no means to adequately represent the forces of repulsion and attraction in models, on the one hand, and the selectivity and asymmetry of interactions, on the other.

Let us consider and compare the known methods of representing relations and interactions, more precisely, forces of attraction and repulsion. Studying the literary sources concerning models of artificial life and artificial chemistry, we can find at least three types of such methods:

- based on adjacency and proximity in cellular automata space, e. g. [9, 10];
- based on the use of common objects, primarily data structures, e.g [3, 8];
- based on the rewriting and composition of functions, e. g. [11].

The oldest and most obvious method of representing the forces of attraction came from the theory of cellular automata [9, 10]. This method is based on the proximity and neighbourhood factors. The closer the neighbour is to the cell, the stronger its influence on the cell. The most influential cells are adjacent cells: they can participate in mutual state transformations (reactions) according to the transition rules. The big disadvantage of this method is the need for centralized management to choose which of the possible transition rules should be preferred, or more precisely, which sequence of rule execution should be performed. This management is performed by central processor unit (CPU). Thus, execution of the rules is not arbitrary, although in general selection of rules is equiprobable, and the strength and asymmetry of the rules are not defined. The repulsion force is not represented at all. But, as it was stated earlier, it is preferable that interactions occur spontaneously, due to the properties inherent in the interacting analogues of the molecules, that is, the programs. Thus, the use of proximity and neighbour factors is not quite suitable.

The second group of methods for representing and establishing relationships is based on the use of common objects: data structures or labels. The overview and analysis of such methods is given, for example, in [3]. But again we are faced with the same problem: the need for a decision-making centre that must authorize interaction and supply it with a model analogue of energy. The only energy supply centre is, as it was stated before, the CPU. In [3, 8], it was revealed that the object methods of representing relations are weak, indirect, and hardly adequate. They also suffer from a lack of selectivity and asymmetry. And again, there is no explicit representation of the repulsive force.

The third way to represent attraction between programs, which are understood as artificial molecules, is based on rewriting their source codes and constructing compositions. In [11], the interaction between programs that leads to formation of their composition is initiated by the decision-making centre. This centre arbitrary chooses interacting programs, thereby representing attraction. The repulsive force is not represented.

Let's formulate and clarify the requirements necessary to obtain the desired representation of attraction-repulsion relations in computer models:

- these relationships should occur spontaneously and inevitably;
- relations should be represented through dynamic interactions, not entities;
- a model in general should not have a decision-making centre, an arbitrator, or an initiating energy centre;
- emerging relationships should be direct and stable;
- selectivity and asymmetry are also necessary.

Based on this list of requirements, we return now to the analysis of the model from [8] and its shortcomings, and also determine ways to improve the model.

According to one of our statements, any relation must be represented as a form of dynamics that results from the activity of the interacting parts. The model from [8] was proposed mainly to achieve this. The purpose of this model was to study the possibility of the emergence of dynamic cyclic structures formed from a set of independently functioning processes. These processes unintentionally interacted with each other by transforming data (symbols) stored in separate spatial cells.

All in all, this method is close to the methods of the second group described above, but there are some small but important features. There is no decision-making centre in this model, and it is unpredictable which process will access the data cell first. Also, in this model, there are no explicitly defined "gluing" common objects, such objects appear unintentionally.

These features allow studying self-organization of connections between unspecific common-purpose application programs. This is very important for software development research. But the resulting connections are weak, and the programs in the structures are loosely coupled. The repulsive force is represented implicitly, as a consequence of the inability of the process to establish a connection, which compels the process to explore other spatial cells.

Thus, as indicated in [3], we always face the same set of problems: the lack of spontaneity; the lack of representation of the concepts of force and electric charge; the inadequate representation of the interactions of attraction and repulsion; the inability to represent the asymmetry and selectivity of the interactions. The lack of selectivity and asymmetry implies that two independent processes implementing different functions, for example, f and g, consuming the same data A, have an equal ability to be attracted to the cell containing the data, and there is no means to change this.

To overcome this set of problems, in [12] it was proposed to establish the principles of artificial physics, a new scientific branch for the study of principles and methods that allow reflecting the principles and categories of real-world physics in computer models. By computer models, we mean, first of all, models of artificial life and artificial chemistry.

Also in [12], a method for representing the electric charge was proposed. The non-specificity of programs in model from [8] is both a strong and a weak side. Previously, we outlined the strong side: it is possible to study unintentional formation of structures from the set of randomly chosen applications. But the non-specificity makes connections weak and hinders selectivity. To eliminate this, we must sacrifice the non-specificity and endow programs with "hooks and loops" that allow attraction modelling. This corresponds to the "key-lock" or "induced fit" principles in the real-world chemistry [13].

Thus, programs or artificial molecules should get specific tools that allow them to "hook" another program or "be hooked" by another program. These tools are intended to mimic the action of an electric charge or the interaction of elementary particles in atoms and molecules of the real-world physics and chemistry. Most importantly, the relations between programs should not be represented by tools that are understood as common objects, but by interactions that are performed using such tools. In the real-world chemistry, the chemical composition of substances determines not only the properties of substances, but also the set of possible reactions and their course. Similarly, tools that model "hooks" and "loops" should determine the possible interactions of artificial molecules, i. e. programs.

In [12], it was suggested to use sockets as such tools. These tools are communication tools that enable interactions between processes. Therefore, they are the means to establish relations in the form of dynamics. This is a unique feature of sockets as inter-process communication system (IPC) tools.

According to [12], the "hooks and loops" enabling attraction are understood as "sinks" and "sources" that jointly model an electric charge. The "sources" represent electrons from the outer orbitals that are loosely bound to the nucleus and can be attracted by other atoms or ions. The "sinks" represent free positions in the outer orbitals that can be occupied by additional electrons attracted from other atoms or ions.

Further in this article, we consider a model illustrating formation of bonds between artificial molecules, or programs, which for the simplicity are understood as monatomic molecules. As a result, this will allow subsequent switching to a more adequate understanding of a program as an analogue of a polymer molecule, or more correctly, an enzyme with several centres of polarization.

1. Method

1.1. General concept

A model concept is a quintuple of the following type:

$$V = \langle S, T, A, G, M \rangle,$$

where S — model space; T — model time; A — a set of atoms (monatomic molecules) or symbols in alphabet; G — a set of rules (grammar rules) to produce complex molecules (words, or symbol chains) from monatomic molecules or atoms (i. e. letters, symbols); M — a set of all permitted molecule types (a vocabulary of all possible words).

As in [8], all simulation takes place in two-dimensional closed cellular space $S \subseteq (\mathbf{N}_{0+} \times \mathbf{N}_{0+})$, $\mathbf{N}_{0+}$ here and further is a set of all non-negative integer numbers. The model time, like in [8], is linear unidirectional: $T \subseteq \mathbf{N}_{0+}$.

The full definition of atoms, or monatomic molecules, is

$$f = \begin{cases} \varnothing, \\ f() = f, \\ f(z_1, \ldots, z_n), \; n \in \{1, \ldots, \infty\}. \end{cases}$$

This means that the function can be empty, with null arguments, or with single or multiple arguments.

In general case, a monatomic molecule $f \in A$ is a multiargument function $f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$, where $\mathbf{X} = \langle x_1, x_2, \ldots, x_n \rangle$ is a vector of variables or null-argument functions, $\mathbf{Y} = \langle y_1, y_2, \ldots, y_m \rangle$ is a sequence of slot symbols, $\forall y_i \in \{>, <\}$. This slot symbols represent "sinks" (<) and "sources" (>) necessary to simulate electric charge and attraction-repelling force. Any slot symbol indicates a position where another function could be inserted to create composite function.

The slot symbols provide formation of the complex molecules according to the following rules from G.

1. First rule is an attraction rule. Slots symbols act in the reciprocal way: a function $f(X_1, >, Y_1)$ could be inserted in a function $g(X_2, <, Y_2)$ instead of symbol <, or vice versa. The resulting transformation is written as $f(X_1, > g(X_2, Y_2) <, Y_1)$ or $g(X_2, < f(X_1, Y_1) >, Y_2)$, both forms are mutually dual and technically describe the same thing. Note that both forms $f(X_1, > g(X_2, Y_2) <, Y_1)$ and $g(X_2, < f(X_1, Y_1) >, Y_2)$ represent only data connections between functions, not the order in which they are calculated. In general, the functions are calculated concurrently.

2. The second rule, or the rule of repulsion, establishes that compositions between functions with similar slot symbols, e. g. $f(X_1, >, Y_1)$ and $g(X_2, >, Y_2)$, or $f(X_1, <, Y_1)$ and $g(X_2, <, Y_2)$, are prohibited.

3. The third rule defines the decomposition method. Any complex molecule $f(X_1, > g(X_2, Y_2) <, Y_1)$ or $g(X_2, < f(X_1, Y_1) >, Y_2)$ may spontaneously decay into $f(X_1, >, Y_1)$ and $g(X_2, <, Y_2)$. The direction of the brackets in complex molecules provides the determination of the corresponding slot symbols for the components of decay.

Let us add a definition of the strength of interactions into our model. The strength of interactions is represented by number of slot symbols. The number of given slot symbols can be "neutralized" by the same number of oppositely directed symbols. This means that n slot symbols in function $f(X_1, >_1 \ldots >_n, Y_1)$ can attract $1 \leqslant p \leqslant n$ functions with oppositely directed slot symbols. In the extreme case, this could be done by single function with the same strength of attraction: $g\left(X_2, <_1 \ldots <_n, Y_2\right)$. Such attraction creates mutually dual formulas like $f\left(X_1, >_1 \ldots >_n g\left(X_2, Y_2\right) <_n \ldots <_1, Y_1\right)$ and $g\left(X_2, <_1 \ldots <_n f\left(X_1, Y_1\right) >_n \ldots >_1, Y_2\right)$. Another extreme case occurs when $f\left(X_1, >_1 \ldots >_n, Y_1\right)$ attracts n various independent functions like $h_1\left(X_{11}, <, Y_{11}\right), \ldots, h_n\left(X_{1n}, <, Y_{1n}\right)$, each of which has a single slot symbol <. The resulting complex molecule is $f\left(X_1, < h_1\left(X_{11}, <, Y_{11}\right) >, \ldots, < h_n\left(X_{1n}, <, Y_{1n}\right) >, Y_1\right)$.

A set of all complex molecules composed from A by applying the rules from G is M. The atoms and molecules from A and M respectively are distributed over the model space, and this distribution is changing in time. Hence, the evolution of our artificial chemistry is a mapping $S \times T \to M \cup A$.

Note that the considered resulting complex molecules have a linear or tree-like structure. It is possible to form more complex and universal structures, for example, cyclic ones, but these cases are not considered here.

Cyclic structures can be easily obtained by slightly enhancing this model. However, we do not consider this enhancement here, so as not to exceed the volume of the article and not distract from priority task, which is the reproduction of the forces of attraction and repulsion like those that act between charged particles. Our next work will be devoted solely to the representation of cyclic structures. Now we focus our attention on the general principle of the emergence of dynamic structures.

In the quintuple $V = \langle S, T, A, G, M \rangle$, nearly all components of artificial chemistry are present: $A \cup M$ is a set of all possible molecules; G is a set of rules; $\langle S, T \rangle$ represents the space-time structure of reaction vessel or domain, and the only thing that needs to be added to complete the definition of artificial chemistry is an algorithm for applying rules from G. The aims of this work are to achieve spontaneity of interactions and avoid introducing a decision centre into the model, so we are to assume that the rules are applied arbitrarily. The described model is naturally suitable for parallel systems with a fine-grained structure, where each atom or molecule has its own CPU. In this case, we are dealing with a multitude of independent decision-making centres that possesses their own energy. This makes the process of interaction between atoms and molecules stochastic. Thus, in our artificial chemistry, the algorithm for applying rules from G is an empty set of operators.

Practically, the triple $\langle A, G, M \rangle$ is a grammar, but it also defines an algebra of "sinks" and "sources". This abstract algebra of "sinks-and-sources" is the basis for a more specific computational model that describes the process of forming complex programs from simpler ones. This model is considered and discussed below.

1.2. H_2O model

Here we present a computer model to reproduce the formation of a water molecule from simpler substances such as oxygen and hydrogen. That is why we name this model H_2O. The model is obtained by instantiating our base model V.

As it was mentioned above, the model space S in H_2O model is cellular and two-dimensional. Any cell of the model space S can be empty or occupied by a single atom or molecule. The atoms composing the set A belong to hydrogen-like or oxygen-like types. Molecules form M can be monatomic, diatomic, or more complex.

According to definition of the model V and the algebra of "sinks-and-sources", it is possible to formulate the concept of an artificial atom as a dynamic entity performing a computational function, possessing an analogue of energy (CPU time) and capable of interacting with other similar entities through "sinks" or "sources" represented by slots in the notation of V. Thus, an abstract artificial atom is a combination of a function, the amount of CPU time and a set of slots.

A hydrogen-like atom (hereinafter, for simplicity, "hydrogen" atom) has a single electron in its outer orbit, which can be shared with other artificial atoms of the same or a different kind. Thus, usually it represents "source" of electric charge. To make our model more intricate and interesting, we consider the outer orbital of "hydrogen" atom as not completely filled. There is an empty position there. So, the "hydrogen" atom is both a "sink" and a "source" of electric charge at the same time. In the proposed model, a "hydrogen" atom or a monatomic molecule is light and capable of migration.

An oxygen-like atom (hereinafter, for simplicity, "oxygen" atom) needs two additional electrons to fill the outer orbitals. This atom has a simpler functionality than "hydrogen" and represents only "sink" of charge. Such artificial monatomic molecule is heavier than "hydrogen" and occupies the spatial cell motionlessly.

In addition, as it was proposed in [8], the emptiness of the cell can be considered as a special type of atom, an analogue of an inert gas that is unable to form compounds with other atoms. So we add a special type of atoms $\varnothing$ to A. These atoms perform an empty function; they do not have CPU time portions and slots.

Atoms and molecules can establish relationships provided that they are in the same spatial cell at the same time. This allows atoms and molecules to find a suitable partner by looking for a logical channel or port corresponding to the socket. The connection between atoms or molecules established through a socket can subsequently be broken by any of the interacting partners at any time, or, on the contrary, last indefinitely.

It is obvious that our model combines the characteristics of at least two methods of representing relationships and interactions that were identified earlier: to establish relationships between molecules and atoms, the proximity of reacting partners is necessary; on the other hand, reacting atoms and molecules must have something in common and joint, i. e. a common virtual particle or a model electron.

It is known that in the client-server architecture, the client part plays an active role in establishing interconnections, while the server part is passive and driven. According to this, we consider a negative charge as active and represent it as a client socket. The excess positive charge is represented in the model as a passive and recipient server socket.

The port number plays the role of a principle quantum number that identifies an electron shell or a set of electron orbitals with the same energy level. To establish a connection between artificial atoms, an artificial atom possessing free electrons tries to share them with the other model atoms located in the same spatial cell. In the model H_2O this corresponds to polling of the ports associated with sockets. When searching for a port, the "first response" rule is applied: the number of the first responding port is used.

The port number is associated with both the location, i. e. the coordinates of the spatial cell, and the number of the represented period (orbit) according to the periodic table of artificial elements.

The active client side polls the port numbers in descending order, that is, from the outer orbitals to the inner ones. Therefore, the construction of a model "water" molecule is more probable than the construction of a model diatomic "hydrogen" molecule H_2.

Possible chemical compounds consisting of artificial "hydrogen" and "oxygen" are shown in fig. 1. At the same time, fig. 1 represents graphic language capable to describe combinations of artificial molecules according "sinks-and-sources" algebra. An atom is represented by circle endowed with "sinks" and "sources", which allow to establish connections with other atoms through their "sources" and "sinks" respectively.

Figure 1 demonstrates the tendency present in our artificial model chemistry to form long polymer-like molecules containing chains of "hydrogen" atoms and "hydroxide" groups. Some of these molecules are exotic and do not exist in nature, but in our artificial simplified chemistry, their existence is quite likely.

In our model, we did not aim to fully reproduce the chemistry of water and hydrogen compounds. We are only interested in the process of formation and dynamic transformation of structures, so the compounds in our model are intentionally unstable, and the "hydrogen" atoms are easily detaching and highly volatile. This approach makes it possible to obtain and track the history of many compounds, unlike chemistry with the formation of strong stable bonds. At this stage of research, it is more important to ensure the ability of "atoms" to regroup. If necessary, the stability of the connections can be achieved by slightly changing the parameters of the model.

To study dynamics occurring in the proposed artificial chemistry system, it was necessary to conduct an experiment described below.

2. Experiment description

To conduct an experiment with the previously proposed model H_2O, we created software in Perl. This software consists of several independent programs. The first and most important of them is a modeling tool that implements the previously described model. This main simula-

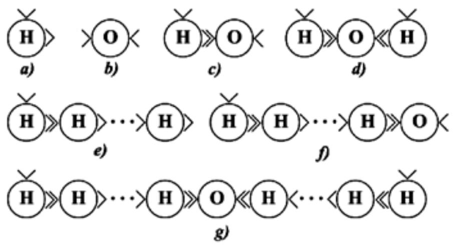


Fig. 1. Artificial atoms and possible molecules in H_2O model:
a — "hydrogen"; *b* — "oxygen"; *c* — "hydroxide" OH; *d* — "water"; *e* — "polymeric hydrogen"; *f* — "polymeric hydrogen with hydroxide group"; *g* — analogues of hydronium type molecules, such as H_1OH_n or H_mOH_n

tion program also generates an initial configuration file and maintains the system logs of events. The remaining programs are auxiliary. They are necessary for analyzing statistics and presenting it in a convenient and understandable form. The simulation begins with the execution of main program. When the main simulation program is terminated, auxiliary programs are started and configuration and log files are analyzed.

The key difference between the main simulation program and our previous development described in [8] is that in this study, the independent processes are not associated with functions and substitution rules, but with artificial atoms. In the model H_2O, we also use sockets as the main means of interprocess communication instead of shared memory in [8].

At the beginning of any experiment, each of independent processes associated with artificial atom of "hydrogen" or "oxygen" goes through initialization phase. At this point, the process creates system of sockets, where passive server sockets are "sinks" and active client sockets are "sources".

Each process is also connected to a location in the cell model space, being associated to randomly assigned cell coordinates (x, y). The process representing "oxygen" is not mobile and is rigidly associated with its current location. On the contrary, the process representing volatile "hydrogen" is mobile and capable of exploring various spatial cells. The "hydrogen" process can change its position by observing its current location in search of molecules and atoms with which to react. Hence, "hydrogen" processes initially are active explorers, while "oxygen" processes are passive responders.

To create effective server socket, we must control the actual number of established and possible connections. To achieve this, we use a solution based on the pre-forking method from [14]. According to this method, the server must maintain a pool of child processes, and each of these processes is waiting for binding to an incoming request from the client. This method has at least two advantages.

First, it is easy to take into account and control both a quantity of child processes and their system identifiers. Knowledge of system identifiers allows controlling the execution and behaviour of child processes by sending commands to them. Knowing the number of executing child processes, it is possible to maintain a pool of ready child processes of constant size. Secondly, the availability of ready child processes significantly increases the performance of the model by reducing the time expected for the creation of child processes. But the price for such advantages is the complication of the model.

The possibility of success of the experiment is based on the principle of port numbering:

$$port\_number = base + aton(concat(x, y, orbit, orbital)),$$

where $port\_number$ — the number of server port; $base$ — a starting point, an arbitrary chosen number; x, y — the coordinates of the spatial cell; $orbit$ — the number of the corresponding atom orbit, $orbital$ — the number of the corresponding atom orbital; $aton$ — a function that converts a string into a number; $concat$ — the concatenation function.

Any port number can simultaneously receive as many incoming requests as there are currently vacant positions available in the corresponding orbit.

As it was previously stated, we used Perl to create modeling software and the main simulation program too. The problem with simulation in Perl is the measuring of time. For Perl programming, the natural and most convenient unit of measurement of time is the second. But the computation speed is much higher, and the same is true for the speed of movement of "hydrogen" atoms in the model, respectively. Therefore, it is necessary to slow down the movement of artificial atoms along the spatial lattice in order to make the measurement of time and speed of movement comparable and so that, as a result, the events of the formation of molecules become observable and distinguishable.

In the case of practical implementation, there is no need to perform such a slowdown. But in the case of an experiment, it is necessary to make sure that our initial assumptions are correct. Namely, that two atoms of "hydrogen" can be attached to one atom of "oxygen" simultaneously, and strictly no more than two.

One of the functions of the main simulation program is event logging. The program produces a set of such event logs. Each log is associated with a "sink" socket belonging to an "oxygen" atom.

Processes spawned by the main program and representing "sinks" implemented as server sockets record information about newly received and already available incoming connections (events) in their logs. This information of events is represented as a pair consisting of a time coordinate and the identifier of the connected client process (i. e., the "hydrogen" atom). In more complex cases, the identifier of the process modeling the "hydrogen" atom may be accompanied by a chain of identifiers of the processes associated with it.

The purpose of the auxiliary programs is to process and present statistical data. The first of these programs explores the temporal sequence of events in spatial cells. To do this, it generates files for each of the spatial cells, where all the events that occurred at a given point in space are indicated for each time coordinate. Owing to this ordering, it is possible to identify the formation of any complex molecule, for example, a "water" molecule when two "hydrogen" atoms join one "oxygen" atom. This approach allows us to study the simulation results from the point of view of the activity of "oxygen" atoms.

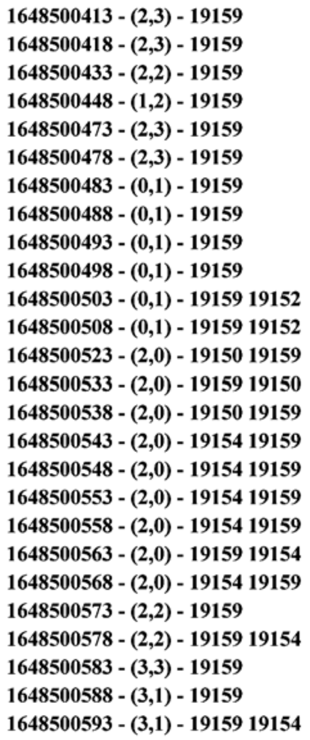
The second program examines statistics from the point of view of "hydrogen" activity. This program processes all log files and creates report files for each "hydrogen" atom present in the model. These reports reflect the trajectories of the mobile "hydrogen" atoms. Any record contains the identifier of the process simulating "hydrogen", the time coordinate and the coordinates of the spatial lattice.

In experiments, the size of spatial lattice varied from 4×4 to 8×8 cells. According a typical scheme, each of the model runs lasted for 30 min, and they were repeated many times with the same results.

The experiment consisted of two parts. In the first part of the experiment, the artificial atom of "hydrogen" implemented only the function of the "source". In the second part of the experiment, the artificial "hydrogen" atom, after joining the "oxygen" atom, also performed the function of a "sink".

In experiments, artificial atoms performed only the simplest operations on the data. The artificial atom of "oxygen" received messages coming through the "sinks" and placed the messages in the log. An artificial atom of "hydrogen" received messages through a "sink" (if there was one), added its own message and transmitted all of them further through a socket simulating a "source".

The results of the experiments are presented in fig. 2 and 3.



```
1648500413 - (2,3) - 19159
1648500418 - (2,3) - 19159
1648500433 - (2,2) - 19159
1648500448 - (1,2) - 19159
1648500473 - (2,3) - 19159
1648500478 - (2,3) - 19159
1648500483 - (0,1) - 19159
1648500488 - (0,1) - 19159
1648500493 - (0,1) - 19159
1648500498 - (0,1) - 19159
1648500503 - (0,1) - 19159 19152
1648500508 - (0,1) - 19159 19152
1648500523 - (2,0) - 19150 19159
1648500533 - (2,0) - 19159 19150
1648500538 - (2,0) - 19150 19159
1648500543 - (2,0) - 19154 19159
1648500548 - (2,0) - 19154 19159
1648500553 - (2,0) - 19154 19159
1648500558 - (2,0) - 19154 19159
1648500563 - (2,0) - 19159 19154
1648500568 - (2,0) - 19154 19159
1648500573 - (2,2) - 19159
1648500578 - (2,2) - 19159 19154
1648500583 - (3,3) - 19159
1648500588 - (3,1) - 19159
1648500593 - (3,1) - 19159 19154
```

Fig. 2. A fragment of log registering movements of process with PID = 19159 (artificial "hydrogen" atom)

```
1651251751 - (1,1) - <H-4397->
1651251753 - (1,1) - <H-4379-> <H-4397->
1651251755 - (1,1) - <H-4397-H-4437->
1651251757 - (1,1) - <H-4380-> <H-4397-H-4441->
1651251759 - (1,1) - <H-4380-H-4401-> <H-4397-H-4441->
1651251787 - (0,1) - <H-4397-> <H-4391->
1651251797 - (1,1) - <H-4397-> <H-4389->
1651251799 - (1,1) - <H-4397->
1651251801 - (1,1) - <H-4397-> <H-4405->
1651251803 - (0,0) - <H-4389-> <H-4397->
1651251803 - (1,0) - <H-4397->
1651251805 - (0,0) - <H-4397->
1651251805 - (1,0) - <H-4397-> <H-4378->
1651251807 - (0,0) - <H-4378-> <H-4397->
```

Fig. 3. A fragment of log registering movements of process with PID = 4397 (a configuration with two "sleeves" is present)

3. Results and discussion

An example of the results for the first part of experiment is shown in fig. 2. This figure shows a fragment of the event log. This event log illustrates the history of the movement of a single artificial atom of "hydrogen". This atom is modeled by a process with the identifier (PID) 19159. The first column represents a time coordinate; the second column represents spatial coordinates (pairs of numbers given in parentheses). The third column contains the PIDs of processes representing "hydrogen" atoms and attached to the same "oxygen" atom in the same spatial cell. One of the PIDs in the sequence is always equal to 19159, since the behavior of this particular process is being investigated.

This fragment of the log proves that spontaneous formation of "hydroxide" (fig. 1, *c*) and "water" (fig. 1, *d*) molecules is possible, and the molecules are quite stable. We see repetitive combinations of {19159, 19154}, {19159, 19150} and {19159, 19152}. From 1648500503 to 1648500508, and from 1648500523 to 1648500568, there were "water" molecules in the cells (0,1) and (2,0), respectively. The chemical composition of these molecules is constant, although the atoms forming them change.

For the second part of experiment, we have modified the form of data output. The conditions of this part of the experiment allow each atom of "hydrogen" attached to an atom of "oxygen" to attract other atoms of "hydrogen". Consequently, the formation of the molecules shown in fig. 1, *f* and fig. 1, *g* is possible under such conditions.

The most complicated configuration of the molecule has become similar to that shown in fig. 1, *g*. This is a configuration consisting of two "sleeves" coming out of a common centre. Each "sleeve" is represented as a sequence of "*H*" symbols denoting "hydrogen" and the PIDs of the corresponding processes. In addition, any of these sequences are enclosed in angle brackets. A log fragment illustrating configurations involving a process with PID = 4397 is shown in fig. 3. It is obvious that at the moment of 1651251759, a configuration with two "sleeves", each of two molecules, has been registered.

Analyzing the results of the experiments, we should note that simple programs equipped with standard interaction mechanisms can spontaneously and unintentionally combine into complex structures. These structures are more obvious and strongly related than those observed in [8]. There are direct explicit connections between simple programs in such structures, unlike those observed in [8].

The attraction between reacting program components is obviously present. Asymmetry and selectivity of interactions are also present in the model, but the repulsive force is still implicitly represented. It is present as the impossibility of attraction.

All types of complex artificial molecules shown in fig. 1 can emerge in the model, with the exception of "polymer hydrogen" (fig. 1, *e*). The reason is the high mobility of "hydrogen" in the model.

In the model, there is one type of atoms that give up electrons ("hydrogen"), and two types of atoms that accept electrons (both "hydrogen" and "oxygen"). This creates an asymmetry towards the predominance of a positive charge. As a result, the artificial model "hydroxide" is obviously positively charged, whereas in nature it is an anion. Hydrogen bonds between molecules are not possible in the model, the obstacle is also related to the problem of charge representation.

In general, this model is only the first approximation on the way to creating artificial chemistry, in which developed program structures can be spontaneously formed from relatively independent components. The model as a whole and the artificial atom model need further improvement.

Conclusions

In this article, we proposed a general concept of an artificial chemistry model capable of reproducing spontaneous interactions between individual programs (functions) performed by independent processes. This concept includes the algebra of "sinks-and-sources" that allow to describe the rules for the self-formation of programs. Compared with λ-calculus [15], this algebra allows forming programs with arbitrary structures.

The proposed general concept leads to the formulation of the concept of an artificial atom and H_2O model, an instance model necessary for conducting experiments in order to prove the possibility of spontaneous self-formation of complex programs from simpler ones.

We have also introduced graphic notation describing the formation of complex molecules from atoms by means of interconnections through "sinks" and "sources". This notation is a graphic representation of the "sinks-and-sources" algebra.

To test our theory, we have developed three types of interacting program components. They come in pure "sink" and "source" types or a hybrid "sink-and-source" type.

Due to the interactions during the experiments, these components spontaneously and unintentionally combined into more complex program structures. The resulting structures can only pump data through themselves, but in the future their functionality may be developed.

The results of the experiments show that the main hypothesis of this article is correct and such spontaneous self-formation of program structures is actually possible under given conditions. The main purpose of the presented study has been achieved.

The results obtained are important for software engineering in order to increase the efficiency of the software development process and to make software flexible and capable of self-improvement during its life cycle. These results are also necessary for research in the fields of artificial life and artificial intelligence to study the self-formation of logical and program structures.

Many models of artificial chemistry are focused on the study and reproduction of reaction systems occurring in the real world and living systems. Examples of such models can be found in [1, 2, 6, 7, 16]. An example of software created on the basis of such a model of artificial chemistry and intended for chemical research is given in chapter 10 in [7].

However, our proposed model is one of the few models focused on the creation of new software engineering paradigms and technologies. The advantage of this model is that it allows compatibility with popular imperative programming tools and can be used for a wide range of applications.

But in general, the proposed models need more refinement. It is also necessary to continue the search for means to represent the repulsive force in a more obvious way. These questions provide material for further research.

References

1. **Dittrich P., Ziegler J., Banzhaf W.** Artificial Chemistries — A Review, *Artificial Life*, 2001, vol. 7, no. 3, pp. 225—275.
2. **Banzhaf W., Yamamoto L.** *Artificial Chemistries,* Cambridge, Massachusetts; London, England, The MIT Press, 2015, 576 p.
3. **Kol'chugina E. A.** *Self-organization and self-development of programs in artificial chemistry models*, Kazan, OOO "Buk", 2019, 256 p. (in Russian).
4. **Eigen M., Schuster P.** *The Hypercycle: A Principle of Natural Self-Organization*, Berlin-Heidelberg-New York, Springer-Verlag, 1979, 98 p.
5. **Anchordoqui L. A., Chudnovsky E. M.** Can Self-Replicating Species Flourish in the Interior of a Star? *Letters In High Energy Physics,* 2020, vol. 2020, LHEP-166, pp. 1—4. DOI: 10.31526/LHEP.2020.166.
6. ***Artificial*** *Life: An Overview* / ed. by C. G. Langton, Cambridge, Massachusetts, The MIT Press, 1997, 336 p.
7. ***Artificial*** *Life Models in Software* / ed. by M. Komosinski, A. Adamatzky, London, Springer, 2009, 462 p.
8. **Kol'chugina E. A.** Spontaneous Emergence of Programs from "Primordial Soup" of Functions in Distributed Computer Systems, *Automatic Control and Computer Sciences*, 2018, vol. 52, no. 1, pp. 40—48. DOI: 10.3103/S0146411618010054.
9. **Von Neumann J.** *Theory of self-replication automata/* ed. by A. W. Burks, Urbana-London, University of Illinois Press, 1966, 416 p.
10. **Mitchell M.** Computations in Cellular Automata: A Selected Review, *Nonstandard Computation*, Weinheim, VCH Verlagsgesellschaft, 1998, pp. 95—140. DOI: 10.1002/3527602968.ch4.
11. **Fontana W.** Algorithmic Chemistry, *Artificial Life II, SFI Studies in the Sciences of Complexity, Vol. X /* ed. by C. G. Langton, C. Taylor, J. D. Farmer, S. Rasmussen, Redwood City, CA, Addison-Wesley, 1991, pp. 159—209.
12. **Kol'chugina E. A.** Model reproduction of non-equilibrium thermodynamics principles as a means to provide software self-development, *IOP Conference Series: Materials Science and Engineering; III International Scientific Conference: Modernization, Innovations, Progress: Advanced Technologies in Material Science, Mechanical and Automation Engineering (MIP-III 2021)*, 29th—30th April 2021, Krasnoyarsk, Russian Federation, 2021, vol. 1155, p. 012054, DOI: 10.1088/1757-899X/1155/1/012054.
13. **Koshland D. E.** Application of a Theory of Enzyme Specificity to Protein Synthesis, *Proceedings of the National Academy of Sciences of the United States of America,* 1958, vol. 44, no. 2, pp. 98—104.
14. **Christiansen T., Torkington N.** *Perl Cookbook, Beijing-Cambridge-Koln-Paris-Sebastopol-Taipei-Tokyo*, O'Reilly & Associates, Inc., 1998. 794 p.
15. **Church A.** An Unsolvable Problem of Elementary Number Theory, *American Journal of Mathematics*, 1936, vol. 58, no. 2, pp. 345—363.
16. **Clark E. B., Hickinbotham S. J., Stepney S.** Semantic closure demonstrated by the evolution of a universal constructor architecture in an artificial chemistry, *Journal of the Royal Society Interface*, 2017, vol. 14, no. 130, article ID 20161033. DOI: 10.1098/rsif.2016.1033.