General Theory of Interaction and Cognitive Architectures

Alexander Letichevsky

Glushkov Institute of Cybernetics, Academy of Sciences of Ukraine
40 Glushkova ave., 03187, Kyiv, Ukraine
let@cyfra.net

Abstract. The challenge of creating a real-life computational equivalent of the human mind is now attracting the attention of many scientific groups from different areas of cybernetics and Artificial Intelligence such as computational neuroscience, cognitive science, biologically inspired cognitive architectures etc. The paper presents a new cognitive architecture based on insertion modeling, one of the paradigms of a general theory of interaction, and a basis for multiagent system development. Insertion cognitive architecture is represented as a multilevel insertion machine which realizes itself as a high level insertion environment. It has a center to evaluate the success of its behavior which is a special type agent that can observe the interaction of a system with external environment. The main goal of a system is achieving maximum success repeated. As an agent this machine is inserted into its external environment and has the means to interact with it. The internal environment of intelligent cognitive agent creates and develops its own model and the model of external environment. If the external environment contains other agents, they can be modeled by internal environment which creates corresponding machines and interprets those machines using corresponding drivers, comparing the behaviors of models and external agents. Insertion architecture is now under development on the base of Insertion modeling system, developed in Glushkov Institute of Cybernetics.

Keywords. AgentBasedSystem, DistributedArtificialIntelligence, Reasoning, FormalMethod, Simulation

Key terms. AgentBasedSystem, DistributedArtificialIntelligence, Reasoning, FormalMethod, Simulation

1 Introduction

General theory of interaction is a theory of information interaction in complex distributed multi-agent systems. It has a long history. Contemporary part of this history can be considered as starting from neuro networks of McCulloch-Pitts [23]. The model of neuro nets caused the appearance of abstract automata theory, a theory which helps study the behavior and interaction of evolving systems independently of their structure. The Kleene-Glushkov algebra [13, 7]

is the main tool for the description of the behaviors of finite state systems. Automata theory originally concentrated on the study of analyses and synthesis problems, generalization of finite state automata and complexity. Interaction in explicit form appeared only in 70s as a general theory of interacting information processes. It includes the CCS (Calculus of Communicated Processes) [24, 25] and the π-calculus of R. Milner [26], CSP (Communicated Sequential Processes) of T. Hoare [10], ACP (Algebra of Communicated Processes) [3] and many other various branches of these basic theories. Now all these calculi and algebras are the basis for modern research in this area. Fairly complete survey of the classical process theory is presented in the Handbook of Process Algebras [4], published in 2001.

Insertion modeling is a trend that is developing over the last decade as an approach to a general theory of interaction of agents and environments in complex distributed multi-agent systems. The first works in this direction have been published in the middle of 90s [6, 15, 16]. In these studies, a model of interaction between agents and environments based on the notion of insertion function and the algebra of behaviors (similar to some kind of process algebra) has been proposed. The paradigm shift from computing to interaction was extensively discussed in computer science that time, and our work was in some sense a response to this trend. But the real roots of the insertion model should be sought even earlier, in a model of interacting of control and operational automata, proposed by V. Glushkov back in the 60s [8, 9] to describe the structure of computers. In the 70s the algebraic abstraction of this model were studied in the theory of discrete processors and provided a number of important results on the problem of equivalence of programs, their equivalent transformations and optimization. Macroconveyor models of parallel computing, which were investigated in 80s years [11], even more close to the model of interaction of agents and environments. In these models, the processes corresponding to the parallel processors can be considered as agents that interact in an environment of distributed data structures.

In recent years, insertion modeling has been applied to the development of systems for the verification of requirements and specifications of distributed interacting systems [2, 12, 19–21]. The system VRS, developed in order from Motorola, has been successfully applied to verify the requirements and specifications in the field of telecommunication systems, embedded systems, and real-time systems. A new insertion modeling system IMS [17], which is under development in the Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine, is intended to extend the area of insertion modeling applications. We found many common features of the tools used in software development area based on formal methods and techniques used in biologically inspired cognitive architectures. This gives us hope to introduce some new ideas to the development of this subject domain.

This paper presents the main principals of insertion modeling and the conception of cognitive architecture based on insertion modeling. To understand the formal part of the paper reader must be familiar with the concepts of labeled

transition system, bisimilarity and basic notions of general process theory. The mathematical foundation of insertion modeling is presented in [18].

2 The Basic Principals

Insertion modeling deals with the construction of models and study the interaction of agents and environments in complex distributed multi-agent systems. Informally, the basic principles of the paradigm of insertion modeling can be formulated as follows.

1. The world is a hierarchy of environments and agents inserted into them.
2. Environments and agents are entities evolving in time.
3. Insertion of agent into environment changes the behavior of environment and produces new environment which is in general ready for the insertion of new agents.
4. Environments as agents can be inserted into higher level environment.
5. New agents can be inserted from external environment as well as from internal agents (environments).
6. Agents and environments can model another agents and environments on the different levels of abstraction.

All these principles can be formalized in terms of transition systems, behavior algebras, and insertion functions. This formalization can be used as high level abstractions of biological entities needed for computer modeling of human mind.

The first and the second principals are commonly used in information modelling of different kinds of systems, for example as in object oriented or agent programming. They are also resembling to M. Minsky's approach of the society of mind [27].

The third principal is clear intuitively, but has a special refinement in insertion modelling. We treat *agents* as transition systems with states considered up to bisimilarity (or up to behavior, which is the same). The *type of an agent* is the set of actions it can perform. The term *action* we use as a synonim of label for transitions, and it can denote signals or messages to send, events in which an agent can participate etc. This is the most general notion of agent which must be distinguished from more special notions of autonomous or intellectual agents in AI.

Transition system consists of states and transitions that connect states. Transitions are labeled by *actions* (signals, events, instructions, statements etc.). Transition systems are evolving in time changing their states, and actions are observable symbolic structures used for communication. We use the well-known notation $s \xrightarrow{a} s'$ to express the fact that transition system can evolve from the state s to s' performing action a. Usually transition systems are nondeterministic and there can be several transitions coming from the same state even labeled by the same action. If we abstract from the structure of states and concentrate only on (branching) sequences of observable actions we obtain the state equivalence

called *bisimilarity* (originated from [28] and [24], exact definition can be found in [18]). Bisimilar states generate the same behavior of transition systems.

Environment by definition is an agent that possesses the *insertion function*. Given the state of environment s and the state of agent u, insertion function computes the new state of environment which is denoted as $s[u]$. Note that we consider states up to bisimilarity and if we have some representation of behaviors, the behaviors of environment and agent can be used as states. The state $s[u]$ is a state of environment and we can use insertion function to insert a new agent v into environment $s[u] : (s[u])[v] = s[u, v]$. Repeating this construction we can obtain the state of environment $s[u_1, u_2, \ldots]$ with several agents inserted into it. Insertion function can be considered as an operator over the states of environment, and if the states are identified with behaviors, then the insertion of a new agent changes the behavior of environment.

Environment is an agent with insertion function, so if we forget the insertion function, then environment can be inserted as agent into a higher level environment and we can obtain hierarchical structure like

$$s[s_1[u_{11}, u_{12}, \ldots]_{E_1}, s_2[u_{21}, u_{22}, \ldots]_{E_2}, \ldots]_E$$

Here notation $s[u_1, u_2, \ldots]_E$ explicitly shows the environment E to which the state s belongs (environment indexes can be omitted if they are known from the context). This refines the fourth principle.

Environment is an agent which can be inserted into external environment and having agents inserted into this environment. The evolution of agents can be defined by the rules for transitions. The rules $s[u] \xrightarrow{a} s[u, v]$ and $s[t[u, v]] \xrightarrow{a} s[t[u], v]$ can be used for the illustration of the 5-th principal.

We consider the creating and manipulation of the models of external and internal environments as the main property of cognitive processes of intellectual agent. Formalization of this property in terms of insertion modeling supports the 6-th principal.

Cognitive architecture will be constructed as a multilevel insertion environment. Below we shall define the main kinds of blocks used for construction of cognitive architecture. They are *local description unites* and *insertion machines*.

3 Multilevel Environments

To represent behaviors of transition systems we use *behavior algebras* (a kind of process algebra). Behavior algebra is defined by the set of actions and the set of behaviors (processes). It has two operations and termination constants. Operations are prefixing $a.u$ (a - action, u - behavior) and nondeterministic choice $u + v$ (u and v - behaviors). Termination constants are successful termination Δ, deadlock 0, and undefined behavior $\bot$. It has also approximation relation $\subseteq$, which is a partial order with minimal element $\bot$, and is used for constructing a complete algebra with fixed point theorem. To define infinite behaviors we use equations in behavior algebra. These equations have the form of recursive definitions $u_i = F_i(u_1, u_2, \ldots), i = 1, 2, \ldots$ and define left hand side functions as

the components of a minimal fixed point. Left hand sides of these definitions can depend on parameters $u_i(x) = F_i(u, x)$ of different types. In complete behavior algebra each behavior has a representation (normal form)

$$u = \sum_{i \in I} a_i.u_i + \varepsilon_i$$

which is defined uniquely (up to commutativity and associativity of nondeterministic choice), if all $a_i.u_i$ are different (ε_u is a termination constant).

The *type of environment* is defined by two action sets: the set of environment actions and the set of agent actions. The last defines the type of agents which can be inserted into this environment: if the set of agent actions is included in the set of agent actions of environment then this agent can be inserted into this environment. This relation is called *compatibility* relation between agents and environments (agent is compatible with environment if it can be inserted into this environment). *Multilevel environment* is a family of environments with distinguished the most external environment. The compatibility relation on the set of environments defines a directed graph and we demand for multilevel environment that the outermost environment would be reachable from any environment of the family in this graph.

To define the insertion function for some environment it is sufficient to define transition relation for all states of environment including states with inserted agents. The common approach is to define behavior by means of rules. The following is an example of such rule:

$$\frac{s \xrightarrow{b} s', u \xrightarrow{a} u'}{s[u] \xrightarrow{c} s'[u']} P(a, b, c)$$

This rule can be interpreted as follows. Agent in the state u can make a transition $u \xrightarrow{a} u'$. Environment allows this transition if the predicate $P(a, b, c)$ is true. This rule defines behavior property of environment in some local neighborhood of the state $s[u]$. So such a rule belongs to the class of local description units discussed in the next section.

At a given moment of time an agent belongs (is inserted) to only one environment. But if the type of an agent is compatible with the type of another environment it can move to this environment. Such a movements can be described by the following types of rules:

$$\frac{u \xrightarrow{moveup\ E} u'}{E[F[u, v], w] \xrightarrow{moveup(F \to E)} E[F[v], u', w]} P_1(E, F, u, moveup(E))$$

moving from internal to external environment;

$$\frac{u \xrightarrow{movedn\ F} u'}{E[F[v], u, w] \xrightarrow{movedn(E \to F)} E[F[u', v], w]} P_2(E, F, u, movedn(F))$$

moving from external environment to internal one;

$$\frac{u \xrightarrow{moveto\ G} u'}{E[F[u,v],G[w]] \xrightarrow{moveto(F \to G)} E[F[v],G[u',w]]} P_3(E,F,u,moveto(F))$$

moving to another environment on the same level. In all cases permitting conditions must include the compatibility conditions for corresponding agents and environments. The rules above define the property of a system called mobility and underlies the calculus of mobile ambients of Luca Cardelli [5].

4 Local Description Units over Attribute Environments

A special type of environments is considered in cognitive architecture to have a sufficiently rich language for the description of environment states properties. These environments are called *attribute environments*. There are two kinds of attribute environments – *concrete* and *symbolic*.

The state of concrete attribute environment is the valuation of *attributes* - symbols that change their values while changing the state in time. Each attribute has type (numeric, symbolic, enumerated, agent and behavior types, functional types etc.). Some of functional and predicate symbols are interpreted symbols. Now logic formulas can be used for the description of properties of agent or environment states. We use the first order logic formulas as the basis that can be extended by fuzzy logic, temporal logic etc.

The general form of *local description unit* is the following:

$$\forall x(\alpha(x,r) \to < P(x,r) > \beta(x,r)),$$

where x is a list of typed parameters, r is a list of attributes, $\alpha(x,r)$ and $\beta(x,r)$ are logic formulas, $< P(x,r) >$ is a process - finite behavior of an environment. Local descriptions can be considered as formulas of dynamic logic, or Hoare triples, or productions - the most popular units of procedural memory in AI. In any case they describe local dynamic properties of environment behavior: for all possible values of parameters, if precondition is true then a process of a local description unit can start and after successful termination of this process a postcondition must be true.

The states of symbolic environment are formulas of basic logic language of environment. Such formulas are abstractions of classes of concrete states. Each symbolic state covers the set of concrete states and the traces generated by local description units cover the sets of concrete traces.

Local description units are the main units of knowledge representation in cognitive architecture. A set of local description units can be used for the definition of transitions of environment. In this case they can be considered as procedural knowledge units. Logic knowledge can be represented as environment with the states representing the current knowledge, and the local description units corresponding to the rules of inference in corresponding calculus. Local description units can be applied in forward and backward modes. Forward mode can be used for the generating of new knowledge, backward mode – for answering queries.

5 Insertion Machines

Another construction blocks for cognitive architecture are insertion machines intended for implementation of insertion environments. The input of insertion machine is the description of a multilevel environment (a model of an environment) and its initial state, an output depends on the goal that is put to machine.

Multilevel environments are represented in cognitive architecture by means of *environment descriptions* for different levels and a set of local description units for insertion functions. Environment description contains the signature of environment that includes types of attributes, types of inserted agents, and also the description of sets of environment and agent actions. Local description units used for the definition of insertion function are organized as a knowledge base with special data structures providing efficient access to the needed descriptions and history of their use.

To implement multilevel environment different kinds of insertion machines are used. But all of them have the general architecture represented on the Fig.1. Three main components of insertion machine are *model driver* (MD), *behavior*

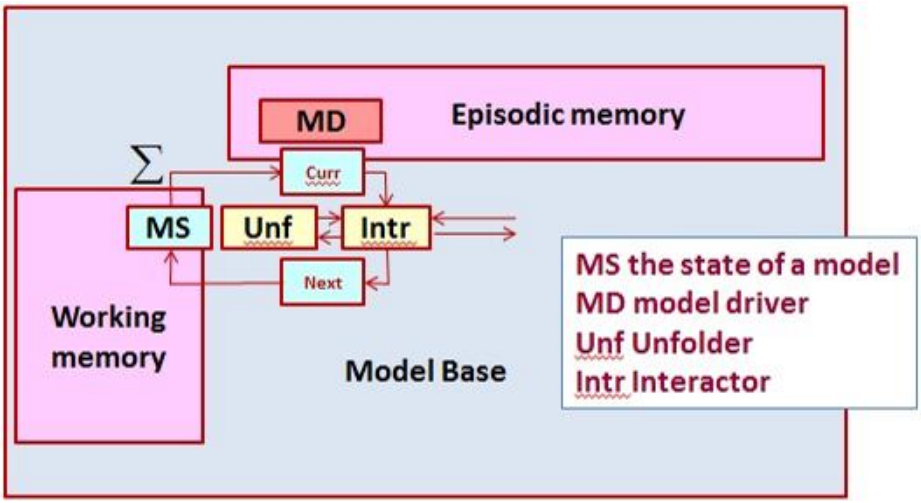

Fig. 1. Architecture of Insertion Machine

unfolder (Unf), and *interactor* (Intr). Model driver is a component which controls the machine traversal along the behavior tree of a model. The state of a model is represented as a text in the input language of insertion machine and is

considered as an algebraic expression. The input language includes the recursive definitions of agent behaviors, the notation for insertion function, and possibly some compositions for environment states. Before computing insertion function the state of a system must be represented in the form $s[u_1, u_2, \ldots]$. This functionality is performed by agent behavior unfolder. To make the movement, the state of environment must be reduced to the normal form

$$\sum_{i \in I} a_i.u_i + \varepsilon$$

where a_i are actions, u_i are environment states, ε is a termination constant. This functionality is performed by the module *environment interactor*. It computes the insertion function calling recursively if it is necessary the agent behavior unfolder.

Two kinds of insertion machines are distinguished: *real time* or *interactive* and *analytical* insertion machines. The first ones are functioning in the real or virtual environment, interacting with it in the real or virtual time. Analytical machines intended for model analysis, investigation of its properties, solving problems etc. The drivers for two kinds of machines correspondingly are also divided into interactive and analytical drivers. Interactive driver after normalizing the state of environment must select exactly one alternative and perform the action specified as a prefix of this alternative. Insertion machine with interactive driver operates as an agent inserted into external environment with insertion function defining the laws of functioning of this environment. External environment, for example, can change a behavior prefix of insertion machine according to their insertion function. Cognitive interactive driver has criteria of successful functioning in external environment, it accumulates the information about its past in episodic memory, develops the models of external environment, uses some learning algorithms to improve the strategy of selecting actions and increase the level of successful functioning. In addition it should have specialized tools for exchange the signals with external environment (for example, perception of visual or acoustical information, space movement etc.).

Analytical insertion machine as opposed to interactive one can consider different variants of making decisions about performed actions, returning to choice points (as in logic programming) and consider different paths in the behavior tree of a model. The model of a system can include the model of external environment of this system, and the driver performance depends on the goals of insertion machine. In the general case analytical machine solves the problems by search of states, having the corresponding properties (goal states) or states in which given safety properties are violated. The external environment for insertion machine can be represented by a user who interacts with insertion machine, sets problems, and controls the activity of insertion machine. Analytical machine enriched by logic and deductive tools are used for generating traces of symbolic models of systems. The state of symbolic model is represented by means of properties of the values of attributes rather than their concrete values.

Insertion machine with separated external environment interface can be implemented as a transition system with hidden structure that separates the kernel

environment state and the states of inserted agents. Such implementation can be more efficient and can be constructed using partial computations or other specialization and optimization programming tools.

6 Cognitive Architecture

Like well-known cognitive architectures such as Soar [14], ACT-R [1] or many other from the list of BICA society [29] insertion cognitive architecture ICAR is an environment for construction of cognitive agents. The main blocks of this architecture are local description units, agents, represented by their behaviors, and insertion machines. Building blocks are collected in memory units that have structures of knowledge bases or associative memories.

On the abstract level ICAR has the same architecture as cognitive agents that can be created in it. From this point of view it can be considered as an intellectual assistant for user who interacts with ICAR in the process of creating cognitive agents. The general architecture of cognitive agent of ICAR is represented on Fig.2.

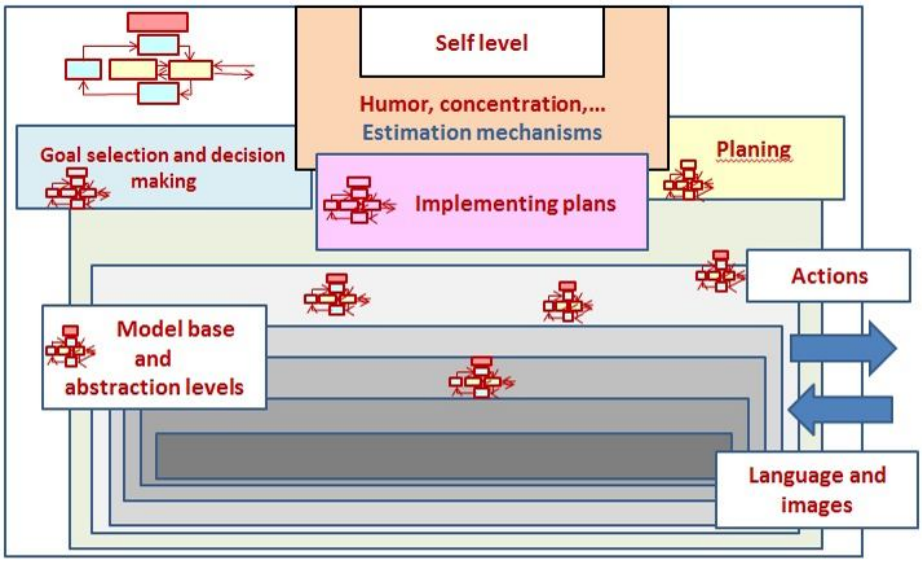

Fig. 2. Insertion cognitive architecture

In general cognitive agent is constructed as a real time multilevel insertion machine which realizes itself as a highest level internal environment. As an agent,

this machine is inserted in its external environment and has the means to interact with it. This external environment includes a user and objects of external (real or virtual) world to which agent has access.

One or several self-models can be inserted into the internal environment of cognitive agent to be used when interacting with external environment or making decisions and planning future activities. An agent has an estimation mechanism to evaluate the success of its behavior. This mechanism is realized in the form of a special agent that can observe the interaction of a system with external environment and make estimation according to some criteria. These criteria can be predefined initially and evolves in the future according to obtained experience. The main goal of a system is achieving maximum success repeated.

The self-models of cognitive agent are created and developed together with the models of external environment. If the external environment contains other agents, they can be modeled by internal environment which creates corresponding machines and interprets those machines using corresponding drivers, comparing the behaviors of models and external agents. All these models are evolving and developing in the process of accumulating the experience in interaction with the external world.

Some mechanisms that model emotional or psychological features (humor and concentration, pleasure and anger, etc.) can be implemented at higher levels of cognitive strucure. The mechanisms of decision making, planning and executing plans are also at higher levels.

The main part of cognitive structure is the base of models describing the history of cognitive agent functioning at different levels of abstraction. The interface with external world provides language (symbolic) communication and image processing. All interaction histories are processed in the working memory of the self-level insertion machines and then transferred to the appropriate levels of a model base.

The model base is always active. The analytical insertion machines which control and manage the structure of model base are always busy with searching solution of problems and performing tasks with ansatisfactory answers, or creating new models. All this activity models subconcious levels of cognition and time-to-time interact with the higher levels of cognitive structure. Independent levels of cognitive structure are working in parallel.

The hierarchy of environments of cognitive agent in some sense are similar to six layers of neocortex. Moving from low levels to higher ones the levels of abstraction are increased and used more and more abstract symbolic models. How to create such models is a big challenge and we are working on it now.

Cognitive analytical insertion machines of ICAR are used by cognitive agents to learn their models and their interaction with external environment to solve problems better, accepts user helps as a teacher and teach user how to interact better with ICAR. General learning mechanisms are the parts of model drivers of different types.

7 Conclusions

The description of cognitive architecture in the last section is a very tentative reflection of our far goals. The nearer goals include the further development of our system of proving pogram correctness [22], communication in natural language, and living in virtual reality. As a zero approximation of ICAR the insertion modeling system [17] is used.

References

1. Anderson, J.R., Lebiere, C.: The Atomic Components of Thought. Mahwah: Lawrence Erlbaum Associates (1998)
2. Baranov, S.,Jervis, C., Kotlyarov, V., Letichevsky, A. and Weigert, T.: *Leveraging UML to deliver correct telecom applications in UML for Real: Design of Embedded Real-Time Systems* by L.Lavagno, G. Martin, and B. Selic (editors), 323-342, Kluwer Academic Publishers (2003)
3. Bergstra, J. A. and Klop J. W.: Process algebra for synchronous communications. Information and Control, 60 (1/3), 109–137 (1984)
4. Bergstra, J. A., Ponce, A. and Smolka, S. A.(eds.): Handbook of Process Algebra. North-Holland (2001)
5. Cardelli, L. and Gordon, A. D.: Mobile ambients. In: Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98, Springer-Verlag (1998)
6. Gilbert, D. R. and Letichevsky, A. A.: A Universal Interpreter for Nondeterministic Concurrent Programming Languages. In: Gabbrielli, M. (Ed.) Fifth Compulog Network Area Meeting on Language Design and Semantic Analysis Methods, September (1996)
7. Glushkov, V.M.: On an Algorithm of Abstract Automata Synthesis. Ukrainian Mthematical Journal, 12(2), 147–156 (1960).
8. Glushkov, V.M.: Automata Theory and Questions of Design Structure of Digital Machines. Cybernetics 1, 3–11 (1965)
9. Glushkov, V.M. and Letichevsky, A. A.: Theory of Algorithms and Discrete processors. In: Tou, J. T. (Ed.) Advances in Information Systems Science, vol. 1, Plenum Press, 1-58 (1969)
10. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall (1985)
11. Kapitonova, J. and Letichevsky, A.: Mathematical Theory of Computational Systems Design. Moscow, Science (1988) (in Russian)
12. Kapitonova, J., Letichevsky, A., Volkov, V. and Weigert, T.: Validation of Embedded Systems. In: R. Zurawski (Ed.) The Embedded Systems Handbook, CRC Press, Miami (2005)
13. Kleene, S. C.: Representation of Events in Nerve Nets and Finite Automata. In: Shannon, C. E., McCarthy, J. (eds.) Automata Studies, Princeton University Press, pp. 3-42 (1956)
14. Laird, J. E., Newell, A., Rosenbloom, P. S.: SOAR: an Architecture for General Intelligence. Artifitial intelligence, 33, 1–64 (1987)
15. Letichevsky, A. and Gilbert, D.: A general Theory of Action Languages. Cybernetics and System Analyses, 1 (1998)

16. Letichevsky, A. and Gilbert, D.: A Model for Interaction of Agents and Environments. In: Bert, D., Choppy, C. and Moses, P. (eds.) Recent Trends in Algebraic Development Techniques. LNCS, vol 1827, Springer Verlag (1999)
17. Letichevsky, A., Letychevskyi, O.and Peschanenko, V.: Insertion Modeling System. In: Proc. PSI 2011, LNCS, vol 7162, pp. 262–274, Springer Verlag (2011)
18. Letichevsky, A.: Algebra of Behavior Transformations and its Applications. In: Kudryavtsev, V. B. and Rosenberg, I.G. (eds.) Structural Theory of Automata, Semigroups, and Universal Algebra. NATO Science Series II. Mathematics, Physics and Chemistry, vol 207, pp. 241–272, Springer Verlag (2005)
19. Letichevsky, A., Kapitonova, J., Letichevsky, A. Jr., Volkov, V., Baranov, S., Kotlyarov, V. and Weigert, T.: Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications. ISSRE 2004, WITUL (Workshop on Integrated reliability with Telecommunications and UML Languages) , Rennes, 4 November (2005)
20. Letichevsky, A., Kapitonova, J., Letichevsky, A. Jr., Volkov, V., Baranov, S., Kotlyarov, V. and Weigert, T.: Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications. Computer Networks, 47, 662–675 (2005)
21. Letichevsky, A., Kapitonova, J., Letichevsky, A. Jr., Volkov, V., Baranov, S., Kotlyarov, V. and Weigert, T.: System Specification with Basic Protocols. Cybernetics and System Analyses, 4 (2005)
22. Letichevsky, A., Letichevsky, O., Morokhovets, M. and Peschanenko, V.: System of Programs Proving. In: Velichko, V., Volosin, A. and Markov, K. (eds.) Problems of Computer Intellectualization, Kyiv, V. M. Glushkov Institute of Cybernetics, pp.133–140 (2012)
23. McCulloch, W.S. and Pitts, W.: A Logical Calculus of the Ideas Immanent in Nervous Activity, Bull. of Math Biophy., 5, 115–133 (1943)
24. Milner, R.: A Calculus of Communicating Systems, LNCS, vol 92, Springer Verlag (1980)
25. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
26. Milner, R.: The Polyadic π-calculus: a Tutorial. Tech. Rep. ECS–LFCS–91–180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK (1991)
27. Minsky, M.: The Society of Mind. Touchstone Book (1988)
28. Park, D.: Concurrency and Automata on Infinite Sequences. LNCS, vol 104, Springer-Verlag (1981)
29. Samsonovich, A. V.: Toward a Unified Catalog of Implemented Cognitive Architectures (Review). In: Samsonovich, A.V., Johansdottir, K.R., Chella, A. and Goertzel, B. (eds.) Biologically Inspired Cognitive Architectures 2010: Proc. 1st Annual Meeting of BICA Society, Frontiers in Artifitial Intelligence and Applications, vol 221, pp. 195–244 (2010)