Computer Science Journal of Moldova, vol.**24**, no.3(72), 2016

Insertion Modeling and Its Applications

Alexander Letichevsky, Oleksandr Letychevskyi
Vladimir Peschanenko

Abstract

The paper relates to the theoretical and practical aspects of insertion modeling. Insertion modeling is a theory of agents and environments interaction where an environment is considered as agent with a special insertion function. The main notions of insertion modeling are presented. Insertion Modeling System is described as a tool for development of different kinds of insertion machines. The research and industrial applications of Insertion Modeling System are presented.

Keywords: process algebra, insertion modeling, formal models, verification.

1 Introduction

Insertion modeling is an approach for research of distributed multi-agent systems and for development of tools for verification of its models. The first papers about insertion modeling were published about 20 years ago [1], [2]. A model of the agents and environments interaction which helps the insertion function notion was presented in these papers.

The main sources of insertion modeling are in a model of interacting control and operating automata, which were found by V.M. Glushkov [3], [4] for the computers description. An algebraic abstraction of this model has been studied in the theory of discrete transformers and has provided some important results on the problem of equivalence of programs, their equivalent transformation and optimization. Macroconveyor models of parallel computing [5] are even closer to the model of interaction between agents and environments. In these

models processes corresponding to parallel processors can be regarded as agents interacting in distributed environment data structures. In recent years the insertion simulation becomes a tool for development applications of verification of systems requirements and specifications of distributed interacting systems [6]–[10].

Another source of insertion modeling is a general theory of interacting information processes, which was created in previous century and is the basis for modern research in this area. It includes CCS (Calculus of Communicated Processes) [11], [12] and π-calculus of R. Milner [13], CSP (Communicated Sequential Processes) of T. Hoar [14], ACP (Algebra of Communicated Processes) [15] and many other different branches of these basic theories. A quite complete review of the classical theory of processes is represented in the handbook on algebra processes [16], which was published in 2001.

The second section is defined by the algebra of behaviors and the bisimulation equivalence of transition systems. The third section introduces the concepts of environment and agents features. The fourth section is devoted to the Insertion modeling system. The fifth section deals with the application of insertion modeling, and finally discusses the possibilities for further development and possible new applications.

2 Behavior algebras

2.1 Transition System

A common approach for describing the dynamics of systems in modern computer science is the notion of transition system, which is defined by sets of states and transitions. Usually this notion is enriched by the additional structures, the most important of which are the transition labelling (labelled transition system introduced by Park [8] to describe the behavior of automata on infinite words). The basic notion in the insertion modeling is an attribute transition system [10], which is defined as follows:

$$< S, A, U, T, \varphi > \tag{1}$$

where S is a set of states, A is a set of actions, which are used for marking the transition, U is a set of labeled attributes, which are used for marking the states, T is transition relation: $T \subseteq S \times A \times S \cup S \times S$, which consists of labeled transitions $s \xrightarrow{a} s'$ and not labeled transitions $s \to s'$.

Function $\varphi : S \to U$ is a function of labeling states. U could be defined as a set $U = D^R$ of mapping of a set R of attributes in a set of data D (a range of values of attributes) or as a $U = \left(D_\xi^{R_\xi}\right)_{\xi \in \Xi}$, where Ξ is a set of data types. A formula of some logic language $L(R)$ is used for symbolic modeling as attributes labels $U \subseteq L(R)$, where R is a set of attributes or a set of attributes with types $R = (R_\xi)_\xi$. It could be interpreted by first order language, which could be expanded by some temporal logic modality. States labeling is considered as some equivalence for symbolic case.

Transition system can also be configured by highlighting some specific sets of states from the set of states S. Among them there are the most important set of initial states S_0, a set of termination states S_Δ and a set of non-defined states $S_\perp$. The last one is used in the theory to determine the relationship of approximation and to build infinite systems in form of finite limits.

As in the theory of automata states the transition systems are considered as some equivalence. In the branch of different equivalences which are considered in the [19] the most important are the trace and bisimulation equivalence (strongest and weakest respectively).

For simplicity, we consider only the system with no hidden transitions. History of operation of attribute transition system is defined as a finite or infinite sequence $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$ of transitions, and a trace corresponding to this history is defined as a sequence $\varphi(s_1) \xrightarrow{a_1} \varphi(s_2) \xrightarrow{a_2} \ldots$

The trace is called maximal if it can't be continued. Let $L(s)$ be the set of all maximal traces which are started in a state s. The states s and s' are called *trace equivalent*, if $L(s) = L(s')$.

Bisimulation equivalence is weaker than trace and defined by thinner manner. A binary relation R on the set of states of the system 1

is called a *relation of bisimulation*, if for every pair (s, s') of its states the following rules are true:

- $(s, s') \in R \Rightarrow \varphi(s) = \varphi(s')$,
- $(s, s') \in R \wedge s \xrightarrow{a} t \Rightarrow \exists t' \left((t, t') \in R \wedge s' \xrightarrow{a} t'\right)$,
- $(s, s') \in R \wedge s' \xrightarrow{a} t' \Rightarrow \exists t \left((t, t') \in R \wedge s \xrightarrow{a} t\right)$.

The states s and s' of the system 1 are called *bisimulation equivalent* if a bisimulation relation R exists, such as $(s, s') \in R$.

Equivalence of systems is usually defined in terms of their equivalence of states. For example, for the initial systems two systems are declared to be equivalent, if the initial state of each of them is equivalent to the initial state of another. The difference between the trace and bisimulation equivalence occurs only in the case of non-deterministic systems. A labeled system is called deterministic if

$$s \xrightarrow{a} s' \wedge s \xrightarrow{a} s'' \Rightarrow \left(s', s''\right) \in R.$$

Two deterministic systems are bisimulation equivalent if and only if they are trace equivalent.

2.2 Behavior algebra

In contrast to the trace equivalence for which the invariant of equivalence (a set of traces) is given together with the definition, the invariant of bisimulation equivalence is not so obvious. In insertion modeling as invariants (generally infinite) expressions or system of equations in algebra behavior are used. A behavior algebra is arranged simply. It is a two-sorted algebra $< U, A >$, the first component U is a set of behaviors, and the second A is a set of actions. The signature of the behavior algebra consists of two operations, one relation and three constants. The first operation $a.u$ is called *prefixing*. Its arguments are action a and behavior u. The result is a new behavior. The second operation is the operation of a non-deterministic choice of $u + v$. This

is a binary operation defined in the set of behaviors. It is commutative, associative and idempotent. The behavior algebras constants are the successful termination Δ, undefined behavior $\perp$ and the deadlock behavior 0, which is a neutral element of non-deterministic choice. On the set of behaviors a binary relation of approximation $\subseteq$ is defined, which is a relation of a partial order with the smallest element Δ. Prefixing and non-deterministic choice operation are monotonous and continuous with respect to this relation. The main role is played by a full behavior algebra $F(A)$, which contains all limits of directed sets and, therefore, a theorem on the minimal fixed point is applied. The exact structure algebra $F(A)$ (for any, including infinite number of actions) is presented in [17].

In the full algebra of behavior each element has the following representation:

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

which is uniquely defined (up to commutativity and associativity), if all $a_i.u_i$ are different.

With each state s of transition system a behavior $beh(s) = u_s$ of system S is associated as the lowest component of the system of equations

$$u_s = \sum_{s \xrightarrow{a} t} a.u_t + \varepsilon_s,$$

where $\varepsilon_s = 0, \Delta, \perp, \Delta + \perp$ depends on the conditions $s \notin S_\Delta \cup S_\perp, s \in S_\Delta \setminus S_\perp, s \in S_\perp \setminus S_\Delta, s \in S_\Delta \setminus S_\perp$, respectively. The main theorem, which characterizes a bisimulation equivalence claims that *two states are bisimulation equivalent if and only if they have equal behavior.* Other approaches to the characterization of a bisimulation equivalence can be found in [20].

3 Agents and Environments

Agent is a transition system, which defines a state up to bisimulation equivalence.

Environment is an agent that has an insertion function. In additional environments there is $< E, C, A, Ins >$, where E is a set of states of an environment, C is a set of actions which could be inserted into an environment, $Ins : E \times F(a) \to E$ is an insertion function. Since the states transition systems are considered as bisimulation equivalence, they can be identified with the behavior and talk about continuity of an insertion function. The main requirement for the environment is a continuity of an insertion function. This assumption implies a number of useful effects. For example, the fact that an insertion function can be set with the help of systems of rewriting rules as the minimal fixed point of the system of functional equations. A result $Ins(e, u)$ of agents insertion, which is in a state u, is defined as $e\,[u]$. Assuming $e\,[u, v] = (e\,[u])\,[v]$ we get the opportunity to talk about the combination of agents that are inserted in an environment and to consider the state of an environment of the form $e\,[u_1, u_2, \ldots]$. Taking into account that an environment is an agent, it can be inserted in a top level environment, considering the multi-level environments like $e\left[e_1\,[u_{11}, u_{12}, \ldots]_{E_1}, e_2\,[u_{21}, u_{22}, \ldots]_{E_2}, \ldots\right]$, where definition $e\,[u_1, u_2, \ldots]_E$ clearly shows environment E, which belongs to the state e. The behavior u of initialized agent defines relation $[u] : E \to T$, which is defined by the relation $[u]\,(e) = e\,[u]$ and an insertional equivalence of agents $\sim_E$ relative to environment E, which is defined by relation $u \sim_E v \Longleftrightarrow [u] = [v]$. This equivalence is usually weaker than bisimulation and plays main role in the applications, because a transformation of algorithms and software implementations of the agents which live in some environment should be executed as transformation which saves insertional equivalence.

In [17] some classification of the insertion functions and the obtained results on a reduction of the complex class of functions to the simple ones are presented.

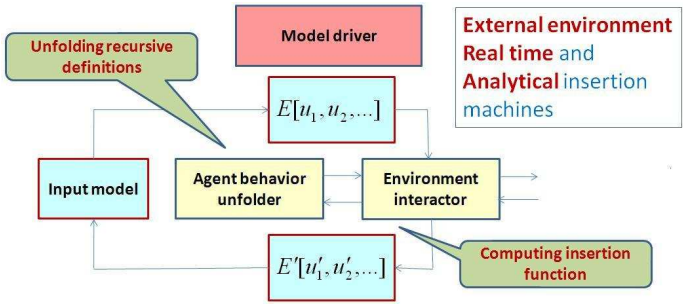

Figure 1. Architecture of Insertion Machine

4 Insertion Modeling System

Insertion modeling system [21] is an environment for the development of insertion machines and performing experiments with them. The notion of insertion machine was used as a tool for programming with some special class of insertion functions. Later this notion was extended for wider area of applications, different levels of abstraction, and multilevel structures.

Insertion model of a system represents this system as a composition of environment and agents inserted into it. Contrariwise the whole system as an agent can be inserted into another environment. In this case we talk about internal and external environment of a system. Agents inserted into the internal environment of a system themselves can be environments with respect to their internal agents. In this case we talk about multilevel structure of agent or environment and about high level and low level environments.

The general architecture of insertion machine is represented in Figure 1.

The main component of insertion machine is model driver, the component which controls the machine movement 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 reduced to the form $E\,[u_1, u_2, ...]$. This functionality is performed by the module called agent behavior unfolder. To make the movement, the state of environment must be reduced to the normal form

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

where a_i are actions, E_i are environment states, ε is a termination constant. This functionality is performed by the module environment interactor. It computes the insertion function calling if it is necessary the agent behavior unfolder. If the infinite set I of indices in the normal form is allowed, then the weak normal form $a.F + G$ is used, where G is arbitrary expression of input language.

Two kinds of insertion machines are considered: *real time* or *interactive* and *analytical* insertion machines. The first ones exist in the real or virtual environment, interacting with it in the real or virtual time. Analytical machines are intended for model analyses, 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.

Analytical insertion machine as opposed to interactive one can consider different variants of making decision 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

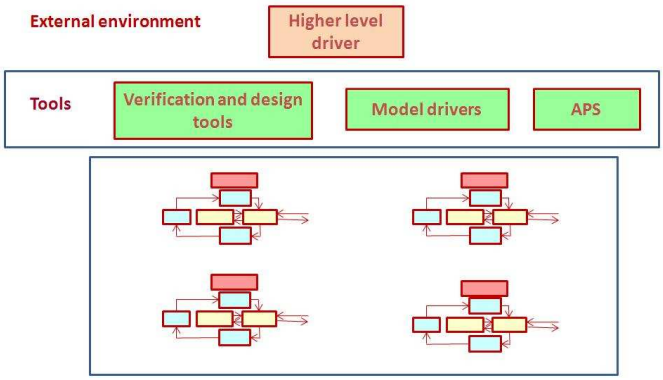

Figure 2. Architecture of Insertion Modeling System

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 is 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.

General architecture of insertion modeling system is represented in Figure 2.

High level model driver provides the interface between the system and external environment including the users of the system. Design tools based on Algebraic Programming system *APS* [21] are used for the development of insertion machines and model drivers for different application domains and modeling technologies. Verification tools are used for the verification of insertion machines, proving their properties statically or dynamically. Dynamic verification uses generating symbolic model traces by means of special kinds of analytical model drivers and deductive components.

The repository of insertion machines collects already developed machines and their components which can be used for the development of new machines as their components or templates for starting. Special library of $APLAN$ functions supports the development and design in new projects. The $C++$ library for IMS supports $APLAN$ compilers and efficient implementation of insertion machines. Deductive system provides the possibility of verification of insertion models [22].

5 Applications

Based on the ideas of insertion modeling the Verification of Requirement Specification (VRS) system was developed by researchers from V.M. Glushkov Institute of Cybernetics of National Academy of Science of Ukraine.

The language of basic protocols is implemented in VRS, which supports the usage of numerical attributes and symbolic types, arrays, lists and functional data types. The deductive system provides proof of the identities in the theory of the first order logic, which is the integration of theories of real and integer linear inequalities, free uninterpreted function symbols and theory query. Symbolic modeling in the VRS is based on satisfiability checking and predicate transformer functions [23].

Proving Programming System is a new and modern system that is designed to maintain a high level of training of qualified specialists in programming. This system is created based on the Insertion Modeling system and Algebraic Programming System which was developed at the V.M. Glushkov Institute of Cybernetics of NAS of Ukraine with the participation of authors from Kherson State University. This system implements Floyds algorithm of proving partial correctness of annotated programs [24].

Insertion Modeling system was successfully used for implementation of theory for building of invariants of the models [25] and loops in software [26], for the set of school computer algebra systems[27], for interleaving reduction in symbolic insertion models [28].

6 Conclusion

In this paper the main notions of insertion modeling are given. Insertion modeling theory is one of the most general theories of process algebra. Its main difference consists in the fact that an environment is considered as an agent with insertion function. Insertion Modeling System was developed for supporting this theory in practice and is used for developing industrial and research insertion machines.

In the nearest future we are planning to use such theory and system for research of models which came from law and economics.

References

[1] D.R. Gilbert and A.A. Letichevsky, "A universal interpreter for nondeterministic concurrent programming languages," In *Fifth Compulog network area meeting on language design and semantic analysis methods*, M. Gabbrielli, Ed. September, 1996. [Online]. Available: http://trove.nla.gov.au/work/7032623.

[2] A. Letichevsky and D. Gilbert, "Interaction of agents and environments," *Recent trends in Algebraic Development technique, LNCS*, vol. 1827, pp. 311–328, September, 2000.

[3] V.M. Glushkov, "The automata theory and design issues digital machines structures," *Cybernetics*, vol. 1, pp. 3–11, 1965. (in Russian)

[4] V. M. Glushkov and A. A. Letichevsky, "Theory of algorithms and descrete processors," *Advances in Information Systems Science (J. T. Tou, Ed.)*, vol. 1, pp. 1–58, Plenum Press, 1969.

[5] M. Kwan, "The design of the ICE encryption algorithm," In *The 4th International Workshop, Fast Software Encryption - FSE '97 Proc. LNCS*, vol. 1267, pp. 69–82, 1997.

[6] Y.V. Kapitonova and A.A. Letichevsky, *The mathematical theory of digital systems*, Moscow, Science, 1988, 295 p. (in Russian)

[7] A. Letichevsky, J. Kapitonova, A. Letichevsky Jr., V. Volkov, S. Baranov, V.Kotlyarov and T. Weigert, "Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications," In *ISSRE 2004, WITUL (Workshop on Integrated reliability with Telecommunications and UML Languages)*, Rennes, November, 2005, pp. 117–132.

[8] S. Baranov, C. Jervis, V. Kotlyarov, A. Letichevsky and T. Weigert, "Leveraging UML to deliver correct telecom applications in UML for Real," In *Design of Embedded Real-Time Systems*, L.Lavagno, G. Martin and B. Selic, Eds. Kluwer Academic Publishers, 2003, pp. 323–342.

[9] J. Kapitonova, A. Letichevsky, V. Volkov and T. Weigert, "Validation of Embedded Systems," In *The Embedded Systems Handbook*, R. Zurawski, Ed. CRC Press, Miami, 2005, pp.6-1–6-26.

[10] A.A.Letichevsky, J.V.Kapitonova, V.A.Volkov, A.A.Letichevsky, jr., S.N.Baranov, V.P.Kotlyarov and T.Weigert, "System Specification with Basic Protocols," *Cybernetics and System Analyses*, vol. 4, 2005, pp. 3–21.

[11] R. Milner, *A Calculus of Communicating Systems*, (LNCS), vol. 92, 1980.

[12] R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

[13] R. Milner, "The polyadic π-calculus: a tutorial," Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK, Tech. Rep. ECSLFCS91180, 1991.

[14] C. A. R. Hoare, "Communicating Sequential Processes," *Prentice Hall*, 1985.

[15] J. A. Bergstra and J. W. Klop, "Process algebra for synchronous communications," *Information and Control*, vol. 60 (1/3), pp. 109–137, 1984.

[16] J. A. Bergstra, A. Ponce and S. A. Smolka, Eds. *Handbook of Process Algebra*, North- Holland, 2001.

[17] A.Letichevsky, "Algebra of behavior transformations and its applications," *Structural theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry*, V.B.Kudryavtsev and I.G.Rosenberg, Eds. vol. 207, pp. 241–272, 2005.

[18] D. Park, "Concurrency and automata on infinite sequences," *LNCS*, vol. 104, 1981, pp. 167–183.

[19] R. J. Glabbeek, *The linear time-branching time spectrum the semantics of concrete, sequential processes*, (Handbook of Process Algebra), J. A. Bergstra, A. Ponce and S. A. Smolka, Eds. North-Holland, 2001.

[20] M. Roggenbach, M. Majster-Cederbaum, "Towards a unified view of bisimulation: a comparative study," *TCS*, vol. 238, pp. 1–130, 2000.

[21] A.A. Letichevsky, O.A.Letychevskyi and V.S. Peschanenko, "Insertion Modeling System," *LNCS*, vol. 7162, pp. 262–274, 2011.

[22] A.A. Letichevsky, O.A. Letychevskyi and V.S. Peschanenko, "Algebraic Programming System APS and Insertion Modeling System IMS, 2016. [Online]. Available: http://www.apsystems.org.ua.

[23] A. Letichevsky, O. Letychevskyi, V. Peschanenko and T. Weigert, "Insertion Modeling and Symbolic Verification of Large Systems," *Lecture Notes in Computer Science*, vol. 9369, pp. 3–18, 2015.

[24] O. Letychevskyi, M. Morokhovets and V. Peschanenko, "System of Provable Programming," *Control Systems and Computers*, vol. 6, pp. 64-71, 2012. (in Russian)

[25] A. Letichevsky, A. Godlevsky, A. Guba, A. Kolchin, O. Letichevkyi and V. Peschanenko, "Usage of Invariants for Sym-

bolic Verification of Requirements," *Risc-Linz report series*, vol. 13-06, pp. 124–124, 2013.

[26] M.S. Lvov, "A Method of Proving the Invariance of Linear Inequalities for Linear Loops," *Cybernetics and Systems Analysis*, vol. 50, no. 4, pp. 643–648, 2014.

[27] *National Projects of Kherson State University*, 2016. [Online]. Available: http://www.kspu.edu/About/DepartmentAndServices/DSAICI/internationalprojects/NationalProjects.aspx?lang=en.

[28] A. Letichevsky, O. Letychevskyi and V. Peschanenko, "An Interleaving Reduction for Reachability Checking in Symbolic Modeling," *In: Proc. 11-th Int. Conf. ICTERI 2015, Lviv, Ukraine*, Ermolayev, V. et al., Eds. May, 2015, pp. 338–353. Available: http://ceur-ws.org/Vol-1356/paper_74.pdf.

Alexander Letichevsky, Oleksandr Letychevskyi, Vladimir Peschanenko

Received October 13, 2016

Alexander Letichevsky
V.M. Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine
40 Glushkov ave., Kyiv, Ukraine, 03187
Phone: +38 (044) 526-00-58
E–mail: aaletichevsky78@gmail.com

Oleksandr Letychevskyi
V.M. Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine
40 Glushkov ave., Kyiv, Ukraine, 03187
Phone:+38 (044) 526-00-58
E–mail: lit@iss.org.ua

Vladimir Peschanenko
Kherson State University of Ukraine
27, 40 rokiv Zhovtnya St., Kherson, Ukraine 73000
Phone: +38 (0552) 32-67-68
E–mail: vladim@ksu.ks.ua