Paper

Integrated analysis of communication protocols by means of PLA formalism

Henrikas Pranevicius

Abstract—Aggregate approach and its possibilities for specification and analysis of computer network protocols are presented. The theoretical basis of the aggregate approach is a piece-linear aggregate (PLA) for formal specification of systems. The advantage of that approach is that it permits to create models both for analysis correctness of specifications and simulation. Some methods that can be used for validation and verification of aggregate specifications are presented also.

Keywords—piece-linear aggregates, ESTELLE/Ag specification language, validation, simulation, communication protocols.

1. Introduction

The stage of formal specification is one of the most important during the design of software of communication protocols. Such formal specification is usually used for analysis and implementation purposes. In the stage of analysis it is necessary to resolve two tasks: analysis of logical correctness and evaluation of the system functioning parameters.

Different mathematical schemes are used for creating formal descriptions of systems, such as: different automate models, Petri-nets, data flow and state transition diagrams, temporal logic technique, abstract communicating methods and other [1, 2].

When a formalization method is chosen, it is desirable that both above mentioned analysis tasks could be resolved on the bases of a single formal description. The aggregate approach has such property and it has been successfully used both for correctness analysis and for simulation of computer network protocols [3–5]. Specification language ESTELLE/Ag and the specifications analysis tool PRANAS-2 have been created on the base of the aggregate method (Ag). There are some differences between ESTELLE/Ag and the ESTELLE standard ISO: the piece-linear aggregate model is used in ESTELLE/Ag. The use of such a model instead of a finite-state automate, which is the formal background of the standard ESTELLE, enables to create models both for validation and simulation. This is possible due to the special structure of the piece-linear aggregate. Apart from the discrete components describing the state of the modules, there are also continuous components to control event-sequences in the module. These continuous components are called operations. By means of operators, sequences of actions are described, the intermediate results of which are invisible on the outside. If such operation sequence is being performed at a given instance of time the corresponding operation is called "active". Thus, an individual module involves two types of events: arrival of an input signal and completion of an active operation. The specification analysis system PRANAS-2 consists of the following software tools: a specification editor, a validation subsystem and a simulation subsystem. The editor provides the capability to create a specification in ESTELLE/Ag. The validation subsystem permits to construct a validation model for the program generating the reachability graph. After completing the construction of the reachability graph, it is possible to verify the following specification characteristics: completeness, deadlock freeness, boundedness, absence of static deadlock, absence of dynamic deadlock, termination.

The same specification changes are carried out when the simulation model is creating. This is necessary in order to define the duration of operations and to introduce additional variables for gathering statistics about the evaluated system parameters.

Section 2 describes the general principles of piece-linear aggregates (PLA) formalism. Methods used for correctness analysis of PLA specification are presented in Section 3. Section 4 illustrates the use PLA formalism for formal specification and integrated analysis of event driven local computer network protocol.

2. General principles of the aggregate approach

In the application of the aggregate approach for system specification, the system is represented as a set of interacting piece-linear aggregates. The PLA is taken as an object defined by a set of states Z, input signals X, and output signals Y. The aggregate functioning is considered in a set of time moments $t \in T$. The state $z \in Z$, the input signals $x \in X$, and the output signals $y \in Y$ are considered to be time functions. Apart from these sets, transition H and output G operators must be known as well.

The state $z \in Z$ of the piece-linear aggregate is the same as the state of a piece-linear Markov process, i.e., $z(t) = \big(\upsilon(t), z_\upsilon(t)\big)$, where $\upsilon(t)$ is a discrete state component taking values on a countable set of values; and $z_\upsilon(t)$ is a continues component comprising of $z_{\upsilon 1}(t)$, $z_{\upsilon 2}(t), \ldots, z_{\upsilon k}(t)$ co-ordinates.

When there are no inputs, the state of the aggregate changes in the following manner:

$$\upsilon(t) = \text{const}, \frac{dz_\upsilon(t)}{dt} = -\alpha_\upsilon,$$

where $\alpha_\upsilon = (\alpha_{\upsilon 1}, \alpha_{\upsilon 2}, \ldots, \alpha_{\upsilon k})$ is a constant vector.

The state of the aggregate can change in two cases only: when an input signal arrives at the aggregate or when a continuous component acquires a definite value. The theoretical basis of piece-linear aggregates is their representation as piece-linear Markov processes.

Aggregate functioning is examined on a set of time moments $T = \{t_0, t_1, \ldots, t_m, \ldots\}$ at which one or several events take place, resulting in the aggregate state alternation. The set of events E which may take place in the aggregate is divided into two non-intersecting subsets $E' = E' \cup E''$. The subset $E' = \{e'_1, e'_2, \ldots, e'_N\}$ comprises classes of events (or simply events) e'_i, $i = \overline{1,N}$ resulting from the arrival of input signals from the set $X = \{x_1, x_2, \ldots, x_N\}$. The class of events $e''_i = \{e''_{ij}, j = 1,2,3,\ldots\}$, where e''_{ij} is an event from the class of events e''_i taking place the ***j***th time since the moment t_0. The events from the subset E' are called external events. A set of aggregate input signals is unambiguously reflected in the subset E', i.e., $X \to E'$. The events from the subset $E'' = \{e''_1, e''_2, \ldots, e''_f\}$ are called internal events, where $e''_i = \{e''_{ij}, j = 1,2,3,\ldots\}$, $i = \overline{1,f}$ are the classes of the aggregate internal events. Here, f determines the number of operations taking place in the aggregate. The events in the set E'' indicate the end of the operations taking place in the aggregate.

The events of the subsets E' and E'' are called the evolutionary events of the aggregate. The main evolution events are sufficient for unambiguous determination of the aggregate evolution. Apart from the basic evolutionary events, auxiliary evolutionary events may be considered, which are simultaneous to the basic ones and determine the start of the operations.

For every class of events e''_i from the subset E'', control sequences are specified $\{\xi_j^{(i)}\}$, where $\xi_j^{(i)}$ – the duration of the operation, which is followed by the event e''_{ij} as well as event counters $\{r(e''_i, t_m)\}$, where $r(e''_i, t_m)$, $i = \overline{1,f}$ is the number of events from the class e''_i taken place in the time interval $[t_0, t_m]$.

In order to determine start and end moments of operation, taking place in the aggregate the so-called control sums $\{s(e''_i, t_m)\}$, $\{w(e''_i, t_m)\}$, $i = \overline{1,f}$ are introduced, where $s(e''_i, t_m)$ – the time moment of the start of operation followed by an event from the class e''_i. This time moment is indeterminate if the operation was not started; $w(e''_i, t_m)$ is the time moment of the end of the operation followed by the event from the class e''_i. In case of no priority operations, the control sum $w(e''_i, t_m)$ is determined in the following way: $w(e''_i, t_m) = s'(e''_i, t_m) + \xi_{r(e''_i, t_m)+1}$, if at moment t_m an operation is taking place, which is followed by the event e_i; in the opposite case $w(e''_i, t_m) = \infty$. The infinity symbol (∞) is used to denote the undefined values of the variables.

Control sums determine only the possibility conditions for the events after the moment t_m, while the event occurrence moments are not determined.

Let us specify the meaning of the co-ordinates of the aggregate state. The discrete component of the state, $\upsilon(t_m) = \{\upsilon_1(t_m), \upsilon_2(t_m), \ldots, \upsilon_p(t_m)\}$, presents the system state:

$$z_\upsilon(t_m) = \{w(e''_i, t_m), w(e''_2, t_m), \ldots, w(e''_f, t_m)\}$$

are control co-ordinates specifying the moment of evolutionary events occurrence.

The control co-ordinate $w(e''_i, t_m)$ corresponds to every each e''_i from the subset of events E'', while always $w(e''_i, t_m) \geq t_m$.

The state co-ordinates $z(t_m)$ can change their values only at discrete time moments $\boldsymbol{t_m}$, $m = 1,2,\ldots$ of event occurrence, remaining fixed in each interval $[t_m, t_{m+1}), m = 0,1,2,\ldots$ where t_0 – the initial moment of system functioning.

When the state of the system $z(t_m)$, $m = 0,1,2,\ldots$, is known, the moment t_{m+1} of the following event is determined by a moment of input signal arrival to the aggregate or by the equation:

$$t_{m+1} = \min\{w(e''_i, t_m)\},\ 1 \leq i \leq f.$$

Class of the next event e_{m+1} is specified by an input signal if it arrives at the time moment t_{m+1} or is determined by the control co-ordinate, which acquire minimal value at the moment t_m, i.e., if $w(e''_i, t_m)$ acquires minimal value, then $e_{m+1} = e''_i$.

The operator H states the new aggregate state:

$$z(t_{m+1}) = H[z(t_m), e_i],\ e_i \in E' \cup E''.$$

The output signals y_i from the set of output signals $Y = \{y_1, y_2, \ldots, y_m\}$ can be generated by an aggregate only at occurrence moments of events from the subsets E' and E''. The operator G determines the content of the output signals:

$$y = G[z(t_m), e_i],\ e_i \in E' \cup E'',\ y \in Y.$$

Further transition and output operators will be denoted $H(e_i)$ and $G(e_i)$.

3. Correctness analysis of aggregate specifications

3.1. Reachable states approach for aggregate model validation

An essence of the reachable states method is a use of the global state which is considered as a joint state of a system after aggregate system composition. A graph of the reachable states is created as oriented one: its nodes stand for global states of the system, its arcs indicate the possible transitions from one state to another. Initial and final states must be specified in working out the graph. The resulting

states graph is used for an analysis of defined properties of a system, as some of them are closely related with the graph structure. The given validation method allows to investigate general properties of a system such as boundedness, absence of redundancy in specification, completeness, absence of static deadlocks, absence of dynamic deadlocks, termination.

3.2. Invariant approach for aggregate model validation

A system invariant (I) is the assertion, which describes correct system functioning and it must remain true in spite of the events taking place and system transition from one state to another.

The essence of the method is as follows: assertions are formulated in relation to the co-ordinates of the aggregate model so as to express the requirements for the system functioning.

On the base of a conceptual model of an analysed system we can describe system functioning by the event sequence, which may be represented by the graph G(V), where V is a set of vertices and $\mathbf{A} = \{a_{ij}\}$ is an adjacency matrix. In this case $V = \{e_1, e_2, \ldots, e_n\}$, where e_i is ith event, n is a number of events. $(e_i e_j) \neq (e_j e_i)$, i.e., the graph is oriented.

The set of states, which the system may enter after the event e1, is called as the ith set of possible states (SS_i – symbolic state). $SS_i = \left\{ z \in Z | (\exists z') \left((z' \in Z) \wedge EP_i(z') \wedge (z = H_i(z', P)) \right) \right\}$, where Z is a set of all possible system states, $EP_i(z')$ is an enabling predicate of the event $\boldsymbol{e_i}$ in the state z', P is a set of probabilistic parameters of the system and H_i is a transition operator determining the new system state when the event e_i occurs.

The system considered being in the symbolic state SS_i only if it is in the state z and $z \in SS_i$. Relying this SS_i definition, every event e_i is related to the symbolic state SS_i, therefore replacing the set of vertices V in the graph G(V) by $V' = \{SS_1, SS_2, \ldots, SS_n\}$ while the adjacency matrix $\mathbf{A}$ remains unchanged. We obtain the graph of symbolic states G(V') which describes the system operation by determining the possible set of states and transitions from one symbolic state to another.

The presented formalization and analysis method will be illustrated by example of specification and integrated analysis of timed protocol with slot reuse.

4. Specification, validation and simulation of event-driven local computer network protocol

4.1. Conceptual model of on event-driven local computer network protocol

There are many computer communication applications requiring high bandwidth and high reliability in operation, which still allow simple and low cost implementation. This type of network exists in robotics, vehicles, homes, etc. These applications set restrictions on the system in terms of usable hardware, cost, and cabling. Such networks are in many cases meant for one special application and not for a general purpose use. The number of stations is generally small compared with typical LAN applications, and the variation in the number of stations is small during the life cycle of the network.

Typical requirements for the media access protocols in these applications are: high reliability of the environment, where the electrical disturbance level is a high scalable bandwidth: self-stabilizing properties; and simplicity combined with low cost of implementation. Solutions based on the existing media access standards do not meet these requirements in many cases.

The protocol described by Sintonen is design to offer high bandwidth while keeping the structure simple. The configuration is a physical bus, where stations form a logical ring. The algorithm is based on the noticeable events on the bus (hence the name event-driven bus protocol). The protocol is distributed, except in the initialization phase. Every station listens to the bus and receives both the destination address and the source address, and stores them in the registers *DA* and *SA* respectively. A station is also capable of sending the bus and detecting the event frame ended. The algorithm for sending and receiving is as follows.

Receiving:
When a station notices it's own address in the *DA* field, it receives the frame.

Sending:
When a station has a frame to send, it waits until it receives the address of it's predecessor in the *SA* of the frame. Then it waits for the event frame ended. After that event, it waits a time period $D', D' \leq 2d$, where d is the end to end delay of, the bus. Then it sends its frame, and waits for a time delay $D', D' > 2d$ to hear the next station begin sending. When this happens, the sending phase is ended. If a station has nothing to send its turn comes, it sends an empty no data frame, a kind of a token, to pass the turn to next station in sequence.

There is one station which initializes the ring, known as the fixed control station. The control station can also detect a failed station and is capable of executing a reconfiguration algorithm to restore the normal operation of the ring.

5. Aggregate specification of on event-driven local computer network protocol

An aggregate schemes of a specification of an analyzed event oriented protocol is depicted in Fig. 1. The aggregates $Station\_0, Station\_1, \ldots, Station\_(n-1)$ depict the stations

which are switched on to the network, and the aggregate *Bus* describes the performance channel. *Station_0* is the controlling one. The signals that are transmitted between the aggregates have also been shown in Fig. 1.

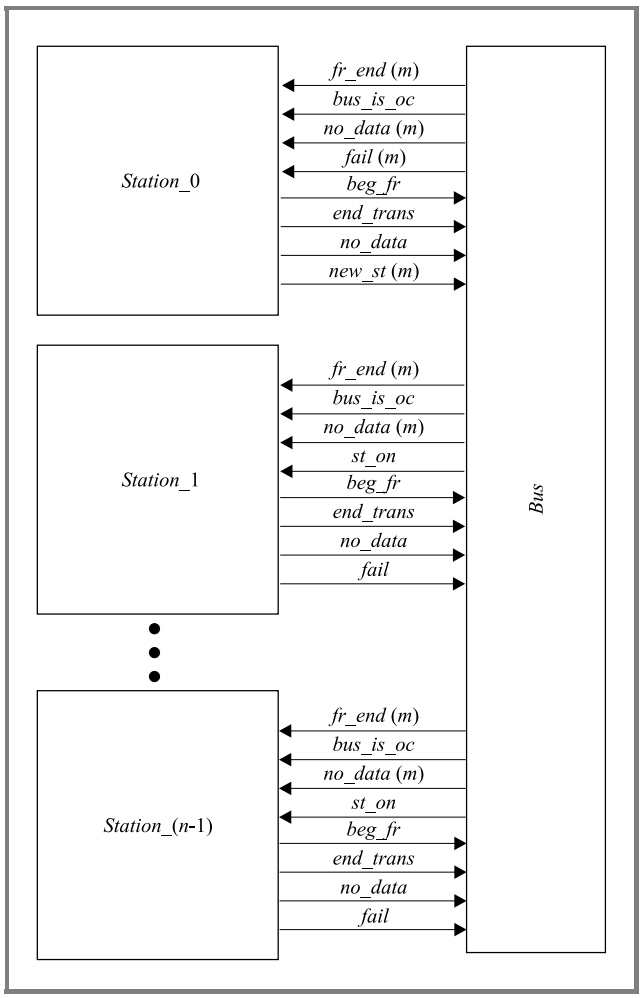

Fig. 1. Aggregate scheme of a model.

Aggregate *Station_nr*, $nr = \overline{1, n-1}$

1. Set of input signal

$$X_{nr} = \{fr\_end(m),\ bus\_is\_oc,\ no\_data(m),\ fail\};$$

where: $fr\_end(m)$ – end of the transmitting; *bus_is_oc* – bus is occupied; *no_data_*(m) – no data for transmission; *st_on* – switching on of the station; n – number o station; m – the number of station where packet is sending.

2. Set of output signals $Y_{nr} = \{y\}$,

$$y \in \{beg\_fr,\ end\_trans,\ no\_data,\ fail\};$$

where: beg_fr – beginning of the frame transmitting; *end_trans* – end of the frame transmitting; *no_data* – no data for transmitting; $fail$ – station is switched off.

3. Set of internal events

$$E''_{nr} = \{e''_1(taim\_DI),\ e''_2(taim\_D),\ e''_3(trans\_fr),$$
$$e''_4(arr\_fr),\ e''_5(swit\_of)\};$$

where: $e''_1(taim\_DI)$ – end of timer *DI*; $e''_2(taim\_D)$ – end of timer *D*; $e''_3(trans\_fr)$ – end of the frame transmitting; $e''_4(arr\_fr)$ – moment of a frame arrival; $e''_5(swit\_of)$ – moment of the station switching.

4. Controlling sequences:

$$e''_i(\ldots) \to \{\xi_{ij}\},\ i = \overline{1, 5},\ j = \overline{1, \infty};$$

where ξ_{ij} – duration of an operation, followed by the event $e''_i(\ldots)$.

5. Discreet component of state

$$\upsilon(t_m) = \{st(t_m),\ actD(t_m),\ sw(t_m)\};$$

where: $st(t_m) \in \{0, 1\}$; 0 – no frame for transmitting, 1 – there is a frame for transmitting;

$$actD(t_m) = \begin{cases} 0, & \text{timer } D \text{ is switched off;} \\ 1, & \text{timer } D \text{ is switched on;} \end{cases}$$

$$sw(t_m) = \begin{cases} 0, & \text{station is switched off;} \\ 1, & \text{station is switched on;} \end{cases}$$

6. Initial state: $st(t_0) = 0$; $act\_D(t_0) = 0$; $sw(t_0) = 0$;
$w\left(e''_1(taim\_DI), t_0\right) = \infty$;
$w\left(e''_2(taim\_D), t_0\right) = \infty$;
$w\left(e''_3(trans\_fr), t_0\right) = \infty$;
$w\left(e''_4(arr\_fr), t_0\right) = t_0 + \xi_{4j}$;
$w\left(e''_5(swit\_of), t_0\right) = t_0 + \xi_{5j}$.

7. Transfer operators:
$H\left(e'(fr\_end)\right)$: (The end of packet sending)
$w\left(e''_1(taim\_DI), t_{m+1}\right) = t_m + \xi_{1j}$
if $sw(t_m) = 1 \wedge m = nr$.
$H\left(e'(bus\_is\_oc)\right)$: (Bus is busy)
$$\left.\begin{array}{l} w\left(e''_2(taim\_D), t_{m+1}\right) = \infty, \\ w\left(e''_4(arr\_fr), t_{m+1}\right) = t_m + \xi_{4j}, \\ act\_D(t_{m+1}) = 0 \end{array}\right\},$$
if $sw(t_m) = 1 \wedge act\_D(t_m) = 1$.
$H\left(e'(no\_data)\right)$: (There are no data for sending)
$w\left(e''_1(taim\_DI), t_{m+1}\right) = t_m + \xi_{1j}$
if $sw(t_m) = 1 \wedge m = nr$;
$$\left.\begin{array}{l} w\left(e''_2(taim\_D), t_{m+1}\right) = \infty, \\ w\left(e''_4(arr\_fr), t_{m+1}\right) = t_m + \xi_{4j}, \\ act\_D(t_{m+1}) = 0 \end{array}\right\},$$
if $sw(t_m) = 1 \wedge act\_D(t_m) = 1$.
$H\left(e''_1(taim\_DI)\right)$: (Timer *DI* has expired)
$$\left.\begin{array}{l} w\left(e''_3(trans\_fr), t_{m+1}\right) = t_m + \xi_{3j}, \\ y = beg\_fr \end{array}\right\},$$
if $st(t_{m+1}) = 1$;

$$\left.\begin{array}{l} w\left(e_2''(taim\_D), t_{m+1}\right) = t_m + \xi_{2j}, \\ act\_D(t_{m+1}) = 1, \\ y = no\_data \end{array}\right\},$$
if $st(t_{m+1}) \neq 1$.

$H\left(e_2''(taim\_D)\right)$: (Timer D has expired)
$y = fail$.

$H\left(e_3''(trans\_fr)\right)$: (The end of packet sending)
$st(t_{m+1}) = 0$;
$w\left(e_2''(taim\_D), t_{m+1}\right) = t_m + \xi_{2j}$;
$act\_D(t_{m+1}) = 1$;
$y = end\_trans$.

$H\left(e_4''(arr\_fr)\right)$: (The packet has arrived)
$st(t_{m+1}) = 1$.

$H\left(e_5''(swit\_of)\right)$: (The station is seething of)
$sw(t_{m+1}) = 0$;
$w\left(e_1''(taim\_DI), t_{m+1}\right) = \infty$;
$w\left(e_2''(taim\_D), t_{m+1}\right) = \infty$;
$w\left(e_3''(trans\_fr), t_{m+1}\right) = \infty$;
$w\left(e_4''(arr\_fr), t_{m+1}\right) = \infty$;
$act\_D(t_{m+1}) = 0$;
$st(t_{m+1}) = 1$.

Aggregate *Station_0*

The functioning of this aggregate is similar to that of the aggregate ***Station_nr***. Therefore, only the differences are presented in respect to the agregate ***Station_nr***.

1. Set of input signals:
   $X_0 = X_{nr} \setminus \{st\_on\} \cup \{fail(m)\}$;
   where: X_{nr} – set of input signal of aggregate *Station_nr*; m – is the number of the stations switched on.

2. Set of output signals:
   $Y_0 = Y_{nr} \setminus \{fail\} \cup \{new\_st(m)\}$;
   where: Y_{nr} – set of output signal of aggregate *Station_nr*; m – the number of the switched on station.

3. Set of internal events:
   $E_0'' = E_{nr}'' \setminus \left\{e_5''(swit\_off),\ e_7''(taim\_T)\right\}$
   $\cup \left\{e_{8i}''(swit\_on), \ldots, e_{8,n-1}''(swit\_on)\right\}$;
   where: $e_7''(taim\_T)$ – end of timer T; $e_{8i}''(swit\_on)$ – ith station switched on.

4. Controlling sequences for the events are introduced $e_7''(\ldots)$ and $e_{8i}''(\ldots)$:
   $e_7''(taim\_T) \mapsto \{T\}$;
   $e_{8i}''(swit\_on) \mapsto \{\xi_{ij}\},\ i = \overline{1, n-1},\ j = \overline{1, \infty}$;
   where: ξ_{8ij} – the operation duration after finishing of which the ith station is switched on; T – the duration of timer T.

5. Discrete component of state
   $\upsilon(t_m) = \left\{st(t_m),\ actD(t_m)\right\}$.

6. Initial state:
   $act\_D(t_0) = 1;\ st(t_m) = 0$;
   $w\left(e_7''(taim\_D),\ t_0\right) = t_0 + T$;
   $w\left(e_{8i}''(swit\_on), t_0\right) = \infty,\ i = \overline{1, n-1}$.

7. Transfer operators:
   $H\left(e'(fr\_end)\right)$: (Bus is busy)
   $$\left.\begin{array}{l} w\left(e''(taim\_DI),\ t_{m+1}\right) = t_m + \xi_{1j}, \\ act\_DI(t_{m+1}) = 1 \end{array}\right\},$$
   if $m = nr$;
   $w\left(e_7''(taim\_T),\ t_{m+1}\right) = t_m + T$,
   if $m \neq nr$.

   $H\left(e'(bus\_is\_oc)\right)$: (Bus is occupied)
   $$\left.\begin{array}{l} w\left(e_7''(taim\_T)\right) = t_m + T \\ w\left(e_2''(taim\_D),\ t_{m+1}\right) = \infty, \\ w\left(e_4''(arr\_fr),\ t_{m+1}\right) = t_m + \xi_{4j} \\ act\_D(t_{m+1}) = 0 \end{array}\right\},$$
   if $act\_D(t_{m+1}) = 1$.

   $H\left(e'(no\_data)\right)$: (There are no data for sending)
   $$\left.\begin{array}{l} w\left(e_1''(taim\_DI),\ t_{m+1}\right) = t_m + \xi_{1j}, \\ w\left(e_7''(taim\_T),\ t_{m+1}\right) = \infty, \\ w\left(e_2''(taim\_D),\ t_{m+1}\right) = \infty, \\ act\_DI(t_{m+1}) = 1, \\ act\_D(t_{m+10}) = 0 \end{array}\right\},$$
   if $m = 0$;
   $$\left.\begin{array}{l} w\left(e_2''(taim\_D),\ t_{m+1}\right) = \infty, \\ w\left(e_4''(arr\_fr),\ t_{m+1}\right) = t_m + \xi_{4j}, \\ act\_D(t_{m+1}) = 0 \end{array}\right\},$$
   if $m = 0 \wedge act\_D(t_m) = 1$;
   $w\left(e_7''(taim\_T), t_{m+1}\right) = t_m + T$.

   $H\left(e'(fail)\right)$: (The station is)
   $w\left(e_{8m}''\left(swit\_on(m)\right)\right) = t_m + \xi_{mj}$.

   $H\left(e_1''(taim\_DI)\right)$: (End of timer DI)
   $act\_DI = 0$;
   $$\left.\begin{array}{l} w\left(e_3''(trans\_fr),\ t_m\right) = t_m + \xi_{3j}, \\ y = beg\_fr \end{array}\right\},$$
   if $st(t_{m+1}) = 1$;
   $$\left.\begin{array}{l} w\left(e_2''(taim\_D),\ t_m\right) = t_m + \xi_{2j}, \\ act\_D(t_{m+1}) = 1, \\ y = no\_date \end{array}\right\},$$
   if $st(t_{m+1}) \neq 1$.

   $H\left(e_2''(taim\_D)\right)$: (The end of timer D)
   $act\_D(t_{m+1}) = 0$;
   $w\left(e_1''(taim\_DI),\ t_{m+1}\right) = t_m + \xi_{1j}$,
   $act\_DI(t_{m+1}) = 1$;
   $w\left(e_4''(arr\_fr),\ t_{m+1}\right) = t_m + \xi_{4j}$;
   $y = fail$.

   $H\left(e_3''(trans\_fr)\right)$: (The transmission of packet has ended)
   $st(t_{m+1}) = 0$;
   $w\left(e''(taim\_D),\ t_{m+1}\right) = t_m + \xi_{2j}$,
   $act\_D(t_{m+1}) = 1$;
   $y = end\_trans$.

   $H\left(e_4''(arr\_fr)\right)$: (The packet has arrived)
   $st(t_{m+1}) = 1$.

$H\left(e''_7(taim\_T)\right)$: (The timer T has expired)
$w\left(e''_1(taim\_DI),\ t_{m+1}\right) = t_m + \xi_{1j}$;
$act\_DI(t_{m+1}) = 1$.

$H\left(e''_{8k}\left(swit\_on(k)\right)\right)$: (The station is switching on)
$y = new\_st(k+1)$.

Aggregate *Bus*

1. Set of input signals:
   $X = \{[beg\_fr,\ end\_trans,\ no\_data,\ new\_st(m)]_0,$
   $[beg\_fr,\ end\_trans,\ no\_data,\ fail]_1, \ldots,$
   $[beg\_fr,\ end\_trans,\ no\_data,\ fail]_{n-1}\}$.
2. Set of output signals:
   $Y = \{[fr\_end(m),\ bus\_is\_oc,\ no\_data(m),\ fail(m)]_0,$
   $[fr\_end(m),\ bus\_is\_oc, st\_on,\ no\_data(m)]_1, \ldots,$
   $[fr\_end(m),\ bus\_is\_oc, st\_on,\ no\_data(m)]_{n-1}\}$.
3. Set of internal events $E'' = \varnothing$.
4. State $\upsilon(t_m) = \{q_i(t_m),\ i = \overline{1,N},\ kan(t_m)\}$;
   where: $q_i(t_m) \in \{1, 2, \ldots, N\}$; $q_i(t)$ – the number of successor for the ith station;
   $kan(t_m) = \begin{cases} 0, \text{ channel is idle;} \\ 1, \text{ channel is occupied.} \end{cases}$
5. Initial state:
   $kan(t_0) := 0$; $i := 1$;
   while $i < n$ do begin $q_i(t_0) := i+1$; $i := i+1$; end.
6. Transfer operators:
   $H\left[e'_{1k}\left(new\_st(p)\right)\right] : k = \overline{2,n}$; (New station)
   $i := p$;
   if $i = n$ then $i := 0$;
   while $q_i(t_m) = 0$ do begin $i := i+1$;
   if $i = n$
   then $i := 0$; end;
   $j := 1$;
   while $q_j(t_m) \neq i+1$ do $j := j+1$;
   $q_j(t_{m+1}) := p$; $q_p(t_{m+1}) := i+1$;
   $y_p := st\_on$.

   $H\left[e'_{2k}(beg\_fr)\right] : k = \overline{1,n}$; (The start of packet sending)
   $kan(t_{m+1}) := 1$;
   for $i := 1$ to n do
   if $i \neq k$ and $q_i(t_m) > 0$ then
   $y_i := bus\_is\_oc$.

   $H\left[e'_{3k}(end\_trans)\right] : k = \overline{1,n}$ (The end of packet transmission)
   $kan(t_{m+1}) := 0$;
   for $i := 1$ to n do
   if $i \neq k$ and $q_i(t_m) > 0$ then
   $y_i := fr\_end\left[q_k(t_m)\right]$.

   $H\left[e'_{4k}(no\_data)\right] : k = \overline{1,n}$ (There are no data for sending)
   for $i := 1$ to n do
   if $i \neq k$ and $q_i(t_m) > 0$ then
   $y_i := no\_data\left[q_k(t_m)\right]$.

   $H\left[e'_{5k}(fail)\right] : k = \overline{2,n}$ (The station is switching of)
   $y_1 := fail[k]$;
   $i := 1$
   while $q_j(t_m) \neq k$ do $i := i+1$;
   $q_i(t_{m+1}) := q_k(t_m)$;
   $q_k(t_{m+1}) := 0$.

5.1. Results of validation and simulation

The correctness of the created specification was investigated by means of protocol analysis system PRANAS-2. This system permitted one to investigate general protocol properties such as: completeness; deadlock freeness; boundedness; cyclic behavior; termination.

Table 1
Example of validation

```
{32}      L:  2 3 1 0
          MO:  1 0 1 1 0 1 3
            Tim_T Arr_fr Taim_DI
          M[1]:  2 0 0 0 1 1
            Swit_of Arr_fr
          M[2]:  3 0 0 0 1 1
            Swit_of Arr_fr

  ↓       Taim_DI in MO
{58}      L:  2 3 1 1
          MO:  1 0 0 1 1 1 3
            Tim_T Arr_fr Trans_fr
          M[1]:  2 0 0 0 1 1
            Swit_of Arr_fr
          M[2]:  3 0 0 0 1 1
            Swit_of Arr_fr

  ↓       Trans_Fr in MO
{104}     L:  2 3 1 0
          MO:  1 1 0 1 0 0 3
            Tim_T Arr_fr Taim_D
          M[1]:  2 0 0 1 1 1
            Swit_of Arr_fr Taim_DI
          M[2]:  3 0 0 0 1 1
            Swit_of Arr_fr

  ↓       Taim_DI in M1
{79}      L:  2 3 1 1
          MO:  1 0 0 1 0 0 3
            Tim_T Arr_fr Taim_D
          M[1]:  2 0 1 0 1 1
            Swit_of Arr_fr Trans_fr
          M[2]:  3 0 0 0 1 1
            Swit_of Arr_fr

  ↓       Trans_fr in M1
{150}     L:  2 3 1 0
          MO:  1 0 0 1 0 0 3
            Tim_T Arr_fr Taim_D
          M[1]:  2 1 0 0 1 0
            Swit_of Arr_fr Taim_D
          M[2]:  3 0 0 1 1 1
            Swit_of Arr_fr Taim_DI

  ↓       Taim_DI in M1
{92}      L:  2 3 1 1
          MO:  1 0 0 1 0 0 3
            Tim_T Arr_fr
          M[1]:  2 0 0 0 1 0
            Swit_of Arr_fr
          M[2]:  3 0 1 0 1 1
            Swit_of Arr_fr Trans_fr
```

In Table 1, some validation results are represented. The numbers included in brackets $\{\ldots\}$ refer to the number of the state. Numbers written after `L`, `MO` and `M[i]` have

the following meanings of discrete and continuous coordinates of state:

```
L:    q1; q2; q3; kan;
MO:   nr; act_D; act_DI; act_T;
      act_trans_fr; st; n_act;
M[i],i=1,2:  nr; act_D;
      act_trans_fr; act_DI;sw;st.
```

5.2. Simulation results

Simulation results are represented in Table 2. The parameters of the model are the following: `Taim_Frame` – duration of frames; `Taim_Head` – duratin of the head of frames; `Taim_D` – duration of the timer *D*; `Taim_DI` – duration of the timer *DI*; `Taim_T` – duration of timer *T*; `V` – velocity of the channel; `n` – number of stations; `Arr_Frame` – parameter of a puasonian input stream; `T_swit_on` and `T_swit_off` – intensity of operations `swit_on` and `swit_off`, which have exponential distributions.
Characteristics of the model: `T_Wait` – the mean value of transmitting a frame including the waiting time; `L_Wait` – mean value of the waiting time; `K_Useful` – coefficient utilization of a channel; `K_Full` – coefficient of full utilization of a channel.

Table 2
Simulation results

Taim_Frame = 800 bit, Taim_Head = 160 bit, Tau_Data = 4 bit, Taim_D = 0.0000025 s, Taim_DI = 0.0000012 s, Taim_T = 100 s, T_swit_on = T_swit_of = 0.				
1. V = 10000000 bit/s, Arr_Frame = 0.001 s				
n	*T_Wait*	*L_Wait*	*R_Useful*	*K_Full*
2	0.00011	0.00001	0.1418	0.8323
4	0.00013	0.00003	0.2806	0.8639
6	0.00016	0.00006	0.4118	0.8939
8	0.00019	0.00010	0.5297	0.9207
10	0.00025	0.00016	0.6304	0.9437
2. V = 50000000 bit/s, Arr_Frame = 0.001 s				
n	*T_Wait*	*L_Wait*	*R_Useful*	*K_Full*
2	0.00002	0.00000	0.0307	0.4639
4	0.00002	0.00001	0.0611	0.4831
6	0.00003	0.00003	0.0917	0.5025
8	0.00003	0.00001	0.1219	0.5217
10	0.00003	0.00001	0.1529	0.5413
3. V = 50000000 bit/s, Arr_Frame = 0.000135 s				
n	*T_Wait*	*L_Wait*	*R_Useful*	*K_Full*
2	0.00002	0.00000	0.0212	0.5921
4	0.00003	0.00001	0.3848	0.6882
6	0.00004	0.00002	0.5399	0.7864
8	0.00006	0.00004	0.6513	0.8569
10	0.00009	0.00007	0.7160	0.8979

6. Conclusions

The presented method of formal specification permits on the base of single specification to carry out validation general and individual properties and simulation. It permits to investigate the analysed system more thoroughly.

References

[1] G. I. Holzmann, "The model checker SPIN", *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, pp. 279–295, 1997.

[2] B. P. Zeigler, *Theory of Modelling and Simulation*. New York: Academic Press, 2000.

[3] H. Pranevicius, "Aggregate approach for specification, validation, simulation and implementation of computer network protocols", in *LNCS*, Berlin: Springer-Verlag, 1991, vol. 502, pp. 433–477.

[4] H. Pranevicius, V. Pilkauskas, and A. Chmieliauskas, "Aggregate approach for specification and analysis of computer network protocols", Technologija, Kaunas University of Technology, 1994.

[5] H. Pranevicius, "Formal specification and analysis of distributed systems", in Lecturer Notes "Applications of AI to Production Engineering", Technologija, Kaunas, 1997, pp. 269–322.

[6] H. Pranevicius, "Formal specification and analysis of distributed systems", *J. Intell. Manuf.*, no. 9, pp. 559–569, 1998.

Henrikas Pranevicius is a Professor of the Kaunas University of Technology and the Head of Business Informatics Department. He is habilituated doctor of Technical Sciences at Ryga Electronic and Computer Technics Institute sinice 1984 and doctor of science from Kaunas Politechnical Institute since 1970. Area of his research activity is: formal specification, validation and simulation of distributed systems including telecommunication and logistic systems. The theoretical background of investigation is piece-linear aggregate formalism, which permits to use the single formal specification for models development both for performance and behaviour analysis.
e-mail: hepran@if.ktu.lt
Kaunas University of Technology
Studentu st 50
LT-51368 Kaunas, Lithuania