Formalized description of cyber-physical systems models using temporary non-deterministic automata

D A Trokoz¹,4, K A Zabrodina², V S Safronova², M D Veselova³, M P Sinev², M A Mitrohin¹, E I Gurin¹ and T Y Pashchenko¹

¹ Penza State Technological University, 1a, Baidukova Avenue/11, Gagarina Street, Penza, 440039, Russian Federation
² Penza State University, 40, Krasnaya Street, Penza, 440026, Russian Federation
³ ITMO University, 49, Kronversky Pr., Saint-Petersburg, 197101, Russian Federation
⁴ E-mail: dmitriy.trokoz@gmail.com

Abstract. The principles of building finite state automata taking into account temporal logic were considered. In this case, time delays of the input symbol are added to the elements of the finite state automaton, which allow the automaton to change its state without producing an output symbol, as well as delays of the output symbol that determine the necessary time interval for the corresponding transition. The main approaches to the definition of the concept of a process, such as “process is a state” and “process is a transition”, which are characterized by the principles of signal processing, taking into account the possibility of saving the state and forming a queue for signal processing, are distinguished. The scientific novelty of this article lies in the approach to the formation of a hierarchical cyber system and the use of temporary SNDA. This method allows to synchronize processes in multithreaded data processing in complex organized, multi-level systems and excludes the possibility of unstable operation, and also improves the performance of the entire complex. As a visual representation of the model of the created principle of temporal non-deterministic automata, a system of grinding and transferring grain in a mill was used.

1. Introduction

When designing complex, multilevel systems, as a rule, there is a need to compile a mathematical model describing the operation of the whole complex, based on finite automata [1].

This approach allows to consider in more detail the mechanism of the system operation, namely the state of all objects and their interaction at a particular point in time, as a result of which you can get a formal description of the product being developed [2].

However, the resulting mathematical model does not always provide for the time component, which is necessary for synchronizing processes in multithreaded data processing [3]. The solution to this problem is the use of temporal finite state automata based on temporal logic, which, in turn, combines the concepts of classical logic and temporal aspects [4].

Within the framework of the presented article, two types of finite automata will be considered: deterministic and non-deterministic [5].

Finite state automata are a sequence (tuple) of five elements described in formula (1).

\[ S = (S, I, O, S_0, \lambda_S) \]
where:
- \( S \) – a finite non-empty set of states with a distinguished initial state \( S_0 \);
- \( I \) – a finite input alphabet;
- \( O \) – a finite output alphabet;
- \( \lambda \) \( S \times I \times S \times O \) – a transitive relation.

The automaton operates in the following sequence: at start it is in the initial state \( s_0 \). After the input signal is received, the function \( \lambda \) is executed, which triggers the transition. It is important to note that the system completely determines the transition to the next state based on the previous state and the input signal. The deterministic automaton is constructed according to the principle: for any \( s_1 \in S \) and \( i \in I \), there is at most one pair \((s_2, o) \in S \times O\) satisfying the condition \((s_1, i, s_2, o) \in \lambda\), which means that for a specified period of time, the automaton can be in only one state, otherwise the automaton is called non-deterministic [6].

Using non-deterministic automata makes it possible to design a mathematical model for a complex with parallel data processing, which allows to increase productivity in complexly organized, multi-level systems [7]. However, parallelization of processes can lead to unstable operation and a chaotic order of actions [8].

To prevent uncontrolled behavior of a computing system, it is necessary to use time constraints on performing transitions from one state to another.

Thus, it is necessary to add elements, reflecting the temporal component, to the standard tuple of the automaton model, namely:

- the delay function of the input symbol \( \Delta_i \) is described in the formula (2):

  \[
  \Delta_i : S \rightarrow S \times (N \cup \{\infty\})
  \]

  This function determines the maximum waiting time for the output symbol for each state. Thus, an input signal that does not arrive for a certain time allows the automaton to change its state without producing an output symbol:

- the delay function of the output symbol \( \sigma_i \) is described in the formula (3):

  \[
  \sigma_i : \lambda \rightarrow (\{0\} \cup N)
  \]

  This function determines the necessary time interval for the corresponding transition, as well as the formation of the output symbol. In addition, an integer constant is currently being generated, which is the maximum value of the timer. The timer value is reset after a specified time period, after which a transition to another state is performed, or at the time the automaton receives an input signal.

Using, where necessary, time delays, it is possible not only to regulate and systematize the processing, but also to maximize the speed of work, the complex performance, provide for time costs and design a functional model of the system that meets the specified criteria. However, it is worth noting that for the current period of time, a formalized description can be made taking into account time only using a deterministic finite automaton, since for non-deterministic automata a correct description of the operation of the system with temporal components has not yet been developed [9].

2. **Criteria for constructing finite automata**

There are criteria for constructing finite automata [10]. The key is to define a process, approaches to which can be divided into the following types:

- a process is a state;
- a process is a transition.

In the first case, the composition of a mathematical model is based on a state, which, in turn, is a specific process in a fixed period of time. After completion of this process, a transition to a new state occurs according to formula (4) with the subsequent launch of a new operation, which implies that:

\[
(s_1, \lambda_s) \in s_2
\]
For the correct operation of the automaton model, two principles of signal processing should be taken into account:

- with / without saving the state at the time of the input signal processing;
- with / without a queue for input signals processing.

In automata with saving the state during the formation of the output signal, the state is fixed until the end of its processing, for example, when the state is processing and an external signal is received at the same time, its processing will not switch to the analysis of the received signal. When using an automaton model without saving the state, the current process will be interrupted.

At the time of the simultaneous receipt of several input signals in the current state, one of the following actions must be performed: either work with each of them in the order of priority as they arrive, or remember the last received signal with its further processing.

Using this approach gives a more detailed formalized description that allows to track and take into account all possible emergency situations when designing the system. However, the construction of this model is a laborious process, and its hardware implementation requires additional financial costs.

In the second case, where “a process is a transition”, indicating that the main operations occur during the transition from one state to another, the following statement from formula (5) is true:

\[(\lambda_{s1}, S) \in \lambda_{s2}\] (5)

Similarly, to the principle “a process is a state” when constructing an automaton, two of the above properties are used. The key difference is that when the state is saved, the output symbol of the previous transition is fixed, which excludes the possibility of storing several input symbols, and thereby creating a queue, therefore, only the last incoming signal is processed.

This method is convenient to use since it facilitates the implementation and understanding of this model. But its use in the design of complexly organized, multi-level systems that require synchronization of parallel data processing is impossible, due to the lack of a detailed description of the sequence of operations [11].

3. Description of method

Based on the above stated criteria for the construction of a finite state automaton, a formalized description of a temporal non-deterministic system of automata was developed, which is a finite state automaton with many states, transitions and signals, each element of which can be described as a separate mechanism consisting of a temporal non-deterministic control automaton and a deterministic operational automaton [12].

The control automaton is non-deterministic, which means that it can be in several states at the same time [13]. In this case, the state is a process that runs for a specified period of time, at the end of which a transition occurs to the output signal, during the formation of which the state is saved until the end of its processing. A signal from a system of automata [14] is input to the state in a control non-deterministic automaton. Moreover, during the simultaneous arrival of signals, queuing does not occur, the last incoming signal is processed.

The time for which the signal is processed can be specified as an integer constant responsible for the delay of the input symbol. The delays of the output symbol are not taken into account, since the developed system uses the principle “process is a state”, which excludes the time spent on signal transmission.

The task of the control automaton is to generate a sequence of actions of the operational automaton [15, 16].

The operating automaton is in only one state in a specified period of time, and it can be interrupted by an external signal transmitted by the control automaton. It is important to note that the temporal component in this automaton is not taken into account, due to the fact that the timer is controlled by the control automaton [17, 18].
4. A mathematical model of the mechanism grinding flour

As an example of the formalization of a mathematical model based on the developed method, the mechanism of the soap dish is used, since this device most clearly represents the synthesis of processes occurring within the framework of modern industrial production.

The functional elements of the “Mill” system are shown in figure 1.

The principle of the mill operation is:

- Grain from the “Bunker 1” through the “Damper 2” is transported to the “Mill 4” with screw “Conveyor 3”. The grinding product is transported to the “Bunker 6” with “Noria 5”. “Damper 2” and “Conveyor 3” are switched on simultaneously (damper response time is 5 seconds), “Mill 4” is turned on 10 seconds after turning on “Conveyor 3”, “Noria 5” is turned on 15 seconds after turning on “Mill 4”. Before turning off the system, it is necessary to remove all grain from the “Conveyor 3”, “Mill 4” and “Noria 5”. After the “Damper 2” is closed, the remaining grain on the “Conveyor 3” will be transferred to the “Mill 4” in 10 seconds. After all the remaining grain is transferred to “Mill 4”, the work of “Mill 4” will have to last another 15 seconds, during which the grinding of the remaining grain will be completed and the grinding product will be transferred to “Noria 5”. Upon completion of “Mill 4” operation, “Noria 5” will have to work another 20 seconds to deliver the remaining grinding product to “Bunker 6”.

It is necessary to provide for the disconnection of the line when filling the “Bunker 6” by the signal of the grain level sensor, or when emptying the “Bunker 1” by the signal of the grain sensor [19].

![Figure 1. Functional elements of the Mill system.](image1)

The first step in the design of the model was the construction of a temporal non-deterministic system of automata, shown in figure 2.

![Figure 2. A mathematical model of nondeterministic time system of automata.](image2)
Symbols of nondeterministic time system of automata are shown in table 1.

Table 1. Symbols of automata states.

| States                        | Input signals          | Output signals          | Time |
|-------------------------------|------------------------|-------------------------|------|
| Sstart – initial state        | –                      | y₀ – initial state      | –    |
| Sbunker1 – check for the      | x – starting the system| y₁ – presence of grain  | –    |
| presence of grain in the      |                        | in the “Bunker 1”       |      |
| “Bunker 1”                    |                        |                         |      |
| Sdamper – operation of the    | x₀ – the “Damper 2”    | y₂ – “Damper 2”         | td – 5 s |
| “Damper 2”                    | opening signal,        |                         |      |
|                               | !x₀ – the “Damper 2”   |                         |      |
|                               | closing signal,        |                         |      |
|                               | x₁ – the “Damper 2”    |                         |      |
|                               | opening signal,        |                         |      |
|                               | !x₅ – the “Damper 2”   |                         |      |
|                               | closing signal,        |                         |      |
| Sconveyor – transportation of | x₀ – the signal to the | y₃ – the activity of the | tc – 10 s |
| grain                        | “Conveyor 3”,          | “Conveyor 3”            |      |
|                               | x₂ – the signal to the |                         |      |
|                               | “Conveyor 3”,          |                         |      |
|                               | !x₁ – the signal to stop|                         |      |
|                               | “Conveyor 3”           |                         |      |
| Smill – grain grinding        | x₀ – the signal to the | y₄ – the activity of the | tm – 15 s |
|                               | “Mill 4”,              | “Mill 4”                |      |
|                               | x₃ – the signal to the |                         |      |
|                               | “Mill 4”,              |                         |      |
|                               | !x₂ – the signal to stop|                         |      |
|                               | “Mill 4”               |                         |      |
| Snoria – transfer of grain    | x₀ – the signal to the | y₅ – the activity of the | tn – 20 s |
|                               | “Noria 5”,             | “Noria 5”               |      |
|                               | x₄ – the signal to the  |                         |      |
|                               | “Noria 5”,             |                         |      |
|                               | !x₃ – the signal to stop|                         |      |
|                               | “Noria 5”              |                         |      |
| Sbunker6 – checking for       | x – the signal of feed | y₆ – fullness of the     | –    |
| fullness of grain in the      | grain                  | “Bunker 6” with grain   |      |
| “Bunker 6”                    |                        |                         |      |
| Send – the final state        | !x₄ – the signal shutdown| y₄ – the system shutdown| –    |

Initially, an external signal is input to the automata system, indicating the start of the system. This signal is transmitted to the “Bunker 1” element of the system, in which the grain availability is checked by an integrated sensor. If “Bunker 1” is “not yet empty”, the sensor simultaneously transmits “x₀” to the elements of the automata system: “Damper 2”, “Conveyor 3”, “Mill 4”, “Noria 5”, otherwise “!x₀” to the “Damper 2”. The mathematical model “Bunker 1”, built using a deterministic automaton, is presented in figure 3.
Symbols of deterministic automaton “Bunker 1” are shown in table 2.

Table 2. Symbols of automata states.

| States        | Input signals                  | Output signals                  |
|---------------|--------------------------------|---------------------------------|
| S₀ – not yet empty | !x – the signal of not yet empty   | y₀ – presence of grain in the “Bunker 1” |
| S₁ – empty    | x – the signal of empty         | y₁ – presence of grain in the “Bunker 1” |

The “Damper 2” receives a signal coming from the automata system. If the signal “x₀” has arrived to the initial state, which is accepted as “Closed”, a transition to the “Opening” state occurs, as well as to the “Opened” state until the timer equal to t seconds elapses. In the “Opening” state, the automaton may stay for t seconds, after which the automatic transition to the “Opened” state takes place. In the case when the automaton is in the “Opened” state, upon receipt of an external signal “x₂”, a transition to the “Closing” state is performed, as well as to the “Closed” state with a delay of t seconds. Similar to the “Opening” state, in the “Closing” state the automaton stays for t seconds, after which the automatic transition to the “Closed” state occurs.

The control non-deterministic temporal automaton of “Damper 2” is shown in figure 4.
Table 3. Symbols of automata states.

| States  | Input signals         | Input signals         |
|---------|-----------------------|-----------------------|
| S₀ – closed | !x₀, x₂, x₃ – the signals of closed, | y₀ – closed |
| S₁ – opening | x₀, !x₃ – the signals of opening | y₁ – opened |
| S₂ – opened | x₀, x₁, !x₂ – the signals of opened | y₂ – opened |
| S₃ – closing | !x₁, x₂ – the signals of closing | y₃ – closed |

When the control automaton is in the “Opening” / “Closing” state, a corresponding signal is transmitted to the operating automaton shown in figure 5.

Symbols of the deterministic automaton of “Damper 2” are in table 4.

Table 4. Symbols of automata states.

| States  | Input signals         | Input signals         |
|---------|-----------------------|-----------------------|
| S₀ₛₐₚ – opening | x – the signal of opening | yₒ$p$ – opened |
| S₀ₛₜ – closing | !x – the signal of closing | yₜₜ – closed |

Transitions and interactions between the states of the control temporal non-deterministic automaton and the deterministic operational automaton “Conveyor 3”, “Mill 4” and “Noria 5” are performed similarly to the transitions of the corresponding “Damper 2” automata, the key differences are the states and processes that occur in them.

Symbols of the other non-deterministic and deterministic automatons are in table 5.

Table 5. Symbols of automata states.

| States  | Conveyor 3 | Mill 4 | Noria 5 | Bunker 6 |
|---------|------------|--------|---------|----------|
| S₀     | does not supply grain | does not grind | does not transport grain | not yet full |
| S₁     | start to supply grain | start to grind grain | start to transport grain | full |
| S₂     | supplying grain | grinding grain | transporting grain | - |
| S₃     | finish to supply grain | finish to grind grain | finish to transport grain | - |
| Sₒₚ   | start | start | start | - |
| Sₒₜ   | stop | stop | stop | - |

The final stage of grinding is getting grain into the “Bunker 6”. The grain availability check sensor is similar to the “Bunker 1” sensor, except for the states: the initial state S₀ is “not yet full”, the final S₁ is “full”.

Figure 5. A mathematical model of the operating automaton.
While “Bunker 6” is in the “not yet full” state, a transition to the initial element of the system is performed.

At the moment when “Bunker 6” enters the “full” state, or “Bunker 1” – “empty” state, the sensor of one of these automata will transmit a signal “!x₀” to the “Damper 2” element. Processing of this signal by each functional element takes place sequentially taking into account the temporal components, since for the correct completion of the system operation it is necessary to exclude the presence of grain in the “Damper 2”, “Conveyor 3”, “Mill 4”, “Noria 5”. At the moment when “Noria 5” enters the state “Does not transport grain”, the Mill system completes its work.

The algorithm of the “Mill” system operation is presented below using the temporal UML diagram shown in figure 6.

Figure 6. Temporary UML diagram of the mill system.

5. Conclusion
In this article, temporal deterministic and non-deterministic finite state automata were considered. In addition, approaches to the definition of the concept of a process affecting the construction of an automaton taking into account the criteria were described in detail, and the principles of signal processing were analyzed.

The scientific novelty of this article lies in the approach to the formation of a hierarchical cyber system and the use of temporary SNDA. This method allows to synchronize processes in multithreaded data processing in complexly organized, multi-level systems, which excludes the possibility of unstable operation, and also increases the entire performance of the complex.

In addition, the developed formalized description can be used to design special computers, discrete devices, and port developed software to hardware such as FPGAs.

Acknowledgments
This work was funded by RFBR in accordance with research project No. 19-07-00516.

References
[1] Minzilia Sagadeeva 2018 On Mathematical Model of Optimal Dynamical Measurement in
Presence of Multiplicative and Additive Effects *International Russian Automation Conference (RusAutoCon)* 1-4

[2] Liguo D, Yunpeng C, Chen W and Yan J 2009 Fuzzy Temporal Logic on Fuzzy Temporal Constraint Networks *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*

[3] Sotnikov A P, Shabaldina N V and Gromov M L 2017 Experiments on Parallel Composition of Timed Finite State Machines *the Institute for System Programming of the RAS* 29(3) 233-46

[4] Ben M 2007 Using Temporal Logic to Analyse Temporal Logic: A Hierarchical Approach Based on Intervals *Journal of Logic and Computation* 17(2) 333-409

[5] Toshimitsu U and Shigemasa T 2015 Nonblocking Supervisory Control of Discrete Event Systems Modeled by Mealy Automata with Nondeterministic Output Functions *IEEE Transactions on Automatic Control* 61(3) 799-804

[6] Ajay K and Anil K V 2015 An Improved Algorithm for the Metamorphosis of Semi-Extended Regular Expressions to Deterministic Finite Automata *The Computer Journal* 58(3) 448-56

[7] Hiroki N Tsutomu S and Munehiro M 2010 A regular expression matching using nondeterministic finite automaton *Eight ACM/IEEE International Conference on Formal Methods and Models for Codesign* 26-8

[8] Taylor S and Taylor R 1990 Parallel processing and real-time data acquisition *IEEE Transactions on Nuclear Science* 37(2) 355-60

[9] Rong Su Jan H van Schuppen and Jacobus E Rooda 2008 Supervisor synthesis based on abstractions of nondeterministic automata *9th International Workshop on Discrete Event Systems* 412-8

[10] Cong L and Jie W 2013 Fast deep packet inspection with a dual finite automata *IEEE Transactions on Computers* 62(2) 310-21

[11] Molnar K, Sujbert L and Peceli G 2003 Synchronization of sampling in distributed signal processing systems *IEEE International Symposium on Intelligent Signal Processing* 21-6

[12] Piterman N 2006 From nondeterministic Buchi and Streett automata to deterministic parity automata *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)* 255-64

[13] Melanie Schuh and Jan Lunze 2014 Feedback control of nondeterministic input/output automata *53rd IEEE Conference on Decision and Control* 6737-43

[14] Shigemasa T and Toshimitsu U 2012 Verification of Codiagnosability for Discrete Event Systems Modeled by Mealy Automata With Nondeterministic Output Functions *IEEE Transactions on Automatic Control* 57(3) 798-804

[15] Hien N V and Soriano T 2002 Identification of a control hybrid automaton: an application *IEEE International Conference on Systems* 6

[16] José R González de Mendivil 2018 Conditions for Minimal Fuzzy Deterministic Finite Automata via Brzozowski's Procedure *IEEE Transactions on Fuzzy Systems* 26(4) 2409-20

[17] Vinarskii E M and Zakharov V A 2018 On the verification of strictly deterministic behavior of timed finite state machines *the Institute for System Programming of the RAS* 30(3) 325-40

[18] Trokoz D A, Martyshkin A I, Salnikov I I and Pashchenko D V 2018 Associative Co-processor on the Basis of Programmable Logical Integrated Circuits for Special Purpose Computer Systems *Global Smart Industry Conference* 1-5

[19] Nikolaevich K N, Aleksandrovich Z S, Nikolaevna P U, Petrovich S M, Vladimirovich B V, Anatoliievich T D, Ivanovich M A and Aleksandorovich K V 2020 Instrumental System of Temporal Analysis of Models of Concurrent Computing Systems Constructed Using Theory of Temporary Finite State Automata *IEEE Moscow Workshop on Electronic and Networking Technologies* 1-8