УДК 004.02
DOI: 10.46548/21vek-2020-0952-0007

МЕТОДИКА ПРЕОБРАЗОВАНИЯ ТЕМПОРАЛЬНОГО КОНЕЧНОГО АВТОМАТА В СП-МОДЕЛЬ

Трокоз Дмитрий Анатольевич, кандидат технических наук,проректор по научной работе
Пензенский государственный технологический университет
(440039, Россия, г. Пенза, проезд Байдукова/ул. Гагарина, 1а/11, e-mail: dmitriy.trokoz@gmail.com)
Бикташев Равиль Айнулович, кандидат технических наук,
профессор кафедры «Вычислительная техника»
Синев Михаил Петрович, кандидат технических наук, доцент кафедры «Вычислительная техника»
Федяшов Марк Сергеевич, студент кафедры «Вычислительная техника»
Шеянов Николай Николаевич, студент кафедры «Вычислительная техника»
Пензенский государственный университет
(440026, Россия, Пенза, улица Красная, 40,
e-mails: bra559620@sura.ru, mix.sinev@gmail.com, mark02042001@mail.ru, shnn1998@gmail.com)

Аннотация. Одним из наиболее популярных механизмов для описания или построения алгоритмов является конечный автомат, сочетающий в себе относительную наглядность и простоту построения. Тем не менее процессы в виду своей физической природы не протекают мгновенно, а длятся определённый промежуток времени. В данном случае можно прибегнуть к использованию разновидности временных конечных автоматов. Наряду с конечными автоматами в современных методах моделирования используют сети Петри – мощную математическую модель, способную представить класс алгоритмов любой сложности. Цель данной статьи заключается в описании алгоритма преобразования временного конечного автомата в сеть Петри, содержащую ингибиторные дуги. Данный способ является универсальным, а, следовательно, становится возможным перехода от конечных автоматов к сетям Петри, когда последние становятся более удобны для моделирования процессов.

Ключевые слова: временные конечные автоматы, сети Петри, преобразование в сети Петри, математическая модель, алгоритм.

CONVERSION METHODOLOGY TEMPORAL FINAL STATE MACHINE INTO PETRI NETS

Trokoz Dmitrii Anatolevich, candidate of technical sciences, vice-rector for research
Penza State Technological University
(440039, Russia, Penza, Baydukov Proyezd / Gagarin St., 1a/11, e-mail: dmitriy.trokoz@gmail.com)
Biktashev Ravil Ainulovich, candidate of technical sciences,
professor of sub-department «Computer Engineering»
Sinev Mikhail Petrovich, candidate of technical sciences,
associate professor of sub-department «Computer Engineering»
Fedyashov Mark Sergeevich, student of sub-department «Computer Engineering»
Sheyanov Nikolay Nikolaevich, student of sub-department «Computer Engineering»
Penza State University
(440026, Russia, Penza, Krasnaya street, 40,
e-mails: bra559620@sura.ru, mix.sinev@gmail.com, mark02042001@mail.ru, shnn1998@gmail.com)

Abstract. Final state machine(FSM) is one of the most popular thing for describing or developing an algorithm, because it easy for perception and building. Nonetheless processes with their physical nature don't pass instantly and last defined time interval. In this case we can use temporal kind of final state machines. Along with final state machine in modern modeling methods are used Petri nets – powerful mathematical model that able to represent the most difficult algorithm types. The purpose of the article is describing of temporal final state machine conversion into an inhibitory Petri nets. The method is universal, that's why it is possible go to Petri nets from final state machines when Petri nets are useful for process modeling.

Keywords: temporal final state machines, Petri nets, conversion into Petri nets, mathematical model, algorithm.

Введение. В настоящее время широкое применение нашла такая модель вычислений [1, 2], как конечные автоматы. Тем не менее, конечные автоматы могут оказаться крайне объемными, иметь большое количество состояний и переходов, которые, в сути своей, дублируются, дабы не нарушать правильность построения конечного автомата. Такая ситуация может наблюдаться при создании конечных автоматов, в основе которых заложена модель параллельных вычислений, или же для конечных автоматов, которые отображают возможное поведение пользователя в некоей программной системе, где состояния – это окна какого-либо приложения, а рёбра – возможные переключения между нынешним окном и последующими.

Таким образом, нам необходима некая математическая модель, которая будет проще для представления состояний и переходов, которые могут дублироваться. Известным решением являются сети Петри [3, 4]. Сети Петри представляют собой двудольный ориентированный граф [5, 6], в котором позициям соответствуют вершины, изображаемые кружками, а переходам - вершины, изображаемые прямоугольниками (рис. 1). Также оригинальные сети Петри были расширены такими элементами как ингибиторные (запрещающие) дуги. Дуги могут быть одновременно инцидентны тем вершинам, которые принадлежат разным классам. В позициях могут размещаться метки (фишки), способные перемещаться по сети через переходы. Выполнением сети Петри управляют количество и распределение фишек сети. Фишки находятся в кружках и управляют выполнением переходов сети [7].

Материалы и результаты исследования. Теория сетей Петри изначально были разработаны для работы с параллельными и асинхронными вычислениями. Основанная в начале 60-х годов немецким математиком К. А. Петри теория в настоящее время содержит большое количество средств анализа, имеющих огромное количество программного обеспечения практически во всех отраслях вычислительной техники и даже вне ее [8].

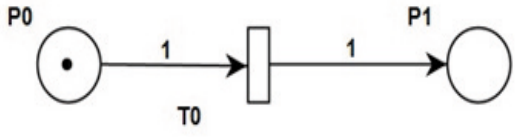

Рисунок 1 – Простейшая сеть Петри

Сети Петри – это мощная математическая модель, которая при правильном использовании может значительно облегчить разработку вычислительных систем любой сложности.

Конвертирование конечного автомата в сеть Петри. В общем случае временной конечный автомат[9] характеризуется состоянием, которое может описывать некий протекающий во времени процесс, и дугами, соединяющими состояния. В реальных системах процесс может длиться определённое время [10]. Для временного автомата введём счётчик, который инкрементируется каждый такт времени *dt* [11]. С каждым состоянием связано число *t*, обозначающее длительность процесса. По появлении входного символа осуществляется переход к соответствующему состоянию, обнуляется счётчик. С этого момента счётчик привязан к активному текущему состоянию [12, 13]. Переход в следующее состояние возможен только после того, как счётчик достиг значения не менее *t*. Для более универсального использования модели введём сигналы, которые могут прервать выполнение текущего процесса и перейти к другому состоянию. Так же скажем, что рассматривать мы будем лишь детерминированная разновидность конечных автоматов.

Такие автоматы представляют из себя множество $M=(S,S_0,X,T,Y,\delta,\lambda)$, где: S - множество внутренних состояний S (внутренний алфавит); S_0- начальное состояние; X - множество входных сигналов X (входной алфавит); T - множество временных меток T; Y - множество выходных сигналов Y (выходной алфавит); δ - функция переходов автомата из одного состояние в другое $\delta: S\times X\rightarrow S$; λ - функция формирования выходных символов $\lambda: S\times X\rightarrow Y$.

Таким образом, задав данные множества и функции, для которых элементы множества выступают в качестве аргументов, мы можем получить временной конечный автомат.

Сеть Петри как математическая модель является множеством $P=(S,T,W,\mu_0)$, где S - конечное множество мест(позиций); T - конечное множество переходов; W - мультимножество дуг (из S в T и из T в S); μ_0 - начальная разметка в сети (расположение фишек);

Последовательно будем рассматривать элементарные составляющие временных конечных автоматов, такие как безусловные переходы, условные переходы по времени, петли и т.д., после чего опишем преобразование временных КА в сети Петри с помощью представленных механизмов.

Безусловный переход. Безусловный переход представляет собой дугу (a_i,a_j). Для состояний a_i и a_j построим эквивалентные состояния p_i и p_j. Для самой дуги построим $T_l \epsilon T$, $w_i=(p_i,T_l)$, $w_j=(T_l,p_j)$. С данным переходом не ассоциировано входных символов, причём, если вершина a_i имеет безусловный переход, то свойство детерминированности конечного автомата гарантирует нам, что это единственное ребро для a_i. Таким образом, добавив во множество σ элемент T_l, соответствующий данному переходу, мы перейдём из p_i в p_j (рис. 2).

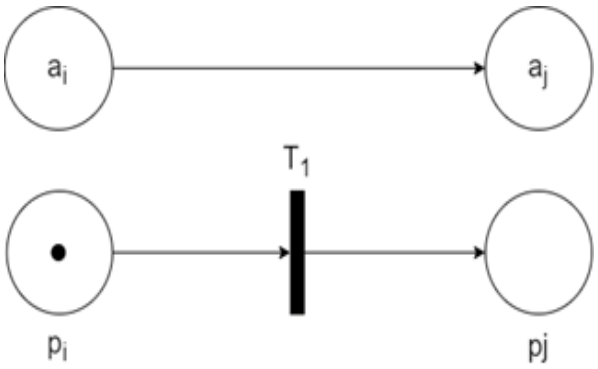

Рисунок 2 – Конвертирование безусловного перехода

Условный переход и петля. Условный переход во временном конечном автомате – это дуга (a_i,a_j) и ассоциированная с ней x_l, где a_i и a_j – начальное и конечное состояние соответственно, z_k – входной символ, при котором осуществляется переход из a_i в a_j. В контексте сетей Петри для a_i и a_j мы построим p_i и p_j – вершины типа «позиция»[14, 15], а вместо дуги постоим $T_l \epsilon T$, $w_i=(p_i,T_l)$, $w_j=(T_l,p_j)$. Также, чтобы отличать условный переход от безусловного, мы введём вершину типа позиция $x_i \epsilon S$. Данная вершина будет иметь фишку в случае, когда подан сигнал x_l (рис. 3).

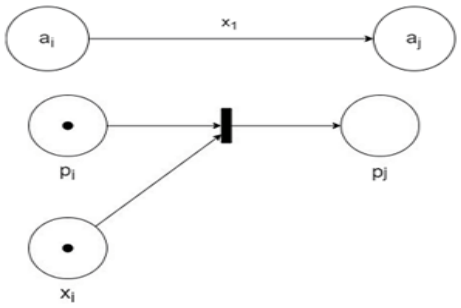

Рисунок 3 – Условный переход

Помимо обычного сигнала x_1 для каждого состояния может существовать его отрицание $!x_i$, по условию которого необходимо перевести автомат в другое состояние [16, 17]. Для этого дополнительно введём переход T_f, и состояние p_k, в которое переходит автомат, когда на его вход поступает символ $!x_i$. Затем построим дуги (p_i, T_f) – обычная дуга и (x_i, T_f) – ингибиторная дуга, которая запрещает переход в случае, если на входной позиции имеются фишки (рис. 4). Таким образом, нахождение фишки в позиции x_i будет соответствовать пришедшему на вход конечного автомата сигналу.

Зачастую возникают ситуации, когда в случае подачи на вход отрицания ожидаемого сигнала нам необходимо вернуться в исходное состояние. В конечных автоматах данную функцию выполняет петля, отличие которой от перехода заключается в том, что $a_i = a_j$. Для конвертирования данного участка мы сделаем следующее: построим дополнительный переход T_f, затем проведём дуги (p_i, T_f) и (x_i, T_f), причём (x_i, T_f) будет являться ингибиторной дугой, которая запрещает переход, если на входных позициях имеются фишки, и разрешает его, если фишки отсутствуют (рис. 5).

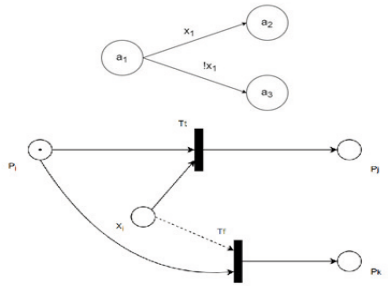

Рисунок 4 – Условный переход по сигналу и его отрицании

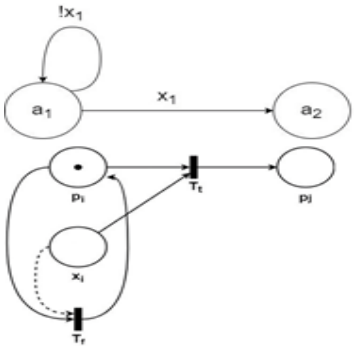

Рисунок 5 – Условный переход с петлёй

Построение условного перехода без использования ингибиторных дуг. Использование ингибиторных дуг может привносить избыточную сложность в сеть Петри, поэтому возможно построить сеть Петри, эквивалентную конечному автомату, без использования запрещающих дуг [18]. Построим дополнительную позицию, обозначающую сигнал $!x_1$, и соединим с переходом T_f. Также соединим переход переходы T_t и T_f с x_1 и $!x_1$ для сохранения сигнала. Таким образом можно преобразовать все ингибиторные дуги в конечном автомате (рис. 6).

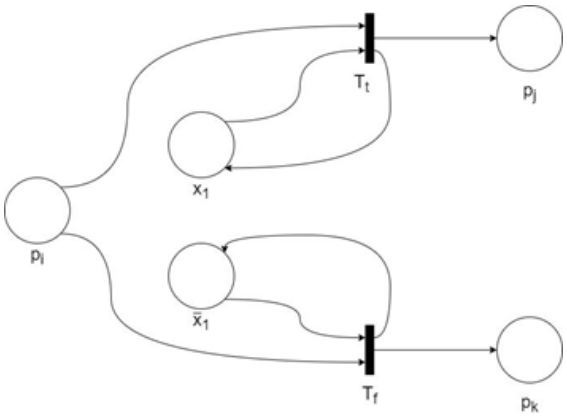

Рисунок 6 – Построение условных переходов без использования ингибиторных дуг

Построение сетей Петри, эквивалентных временному конечному автомату. Прежде всего определим правила взаимодействия внешней системы с данной сетью. Внешняя система может устанавливать фишки в позиции, отвечающие за поступление сигналов x_1, $x_2 \ldots x_n$, а также в позицию *Ptick*, в которую каждый интервал времени *dt* помещается фишка. По поступлении некоторого сигнала x внешняя система помечает фишкой соответствующую позицию в сети (рис. 7).

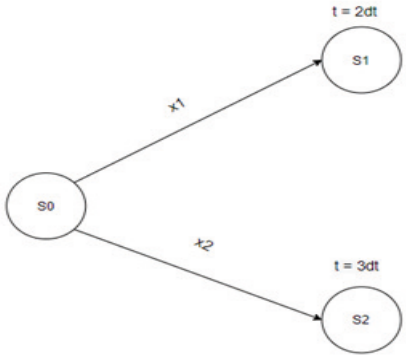

Рисунок 7 – Пример временного конечного автомата

Для построения эквивалентной сети Петри мы определим позицию *pclock*, в которую будут помещаться фишки каждый квант времени *dt*. Также определим позиции *Pend*, характеризующую состояние текущего процесса, S_0 – начальная позиция сети. Изначально сети содержит фишки в состоянии S_0 и *end*.

Построим сеть Петри, эквивалентную следующему конечному автомату: $t_1 = 2dt$, $t_2 = 3dt$, x_1 и x_2 – откладываемые сигналы. Для каждого $s \in S$, $x \in X$ во временном конечном автомате построим $p \in P$, $x \in P$,

si_start∈*T* и *si_end*∈*T*. Переход по отложенному сигналу в следующее состояния возможен в случае, когда предыдущий процесс завершён, поэтому построим дуги (*end, si_start*), (s_0, *si_start*), (x_i, *si_start*), а также ингибиторную дугу (*Pclock, si_start*) во избежание конкуренции за фишку в позиции *end*: возможен случай, когда сигналы x_i и *Pclock* пришли одновременно. *si_start* переводит автомат в новое состояние, а также запускает процесс посредствам высвобождения метки в позиции *end*. Каждый процесс длится определённое время *t=n*dt,n*∈*N*. Поэтому процесс считается завершённым в случае, когда |*Pclock*|=*n*dt*. В соответствии с этим построим дуги (*si, si_end*), (*si_end, si*), (*si_end, end*), а также (*Pclock, si_end*) в количестве n_i. Таким образом, мы получим сети Петри, представленную на рисунке 8.

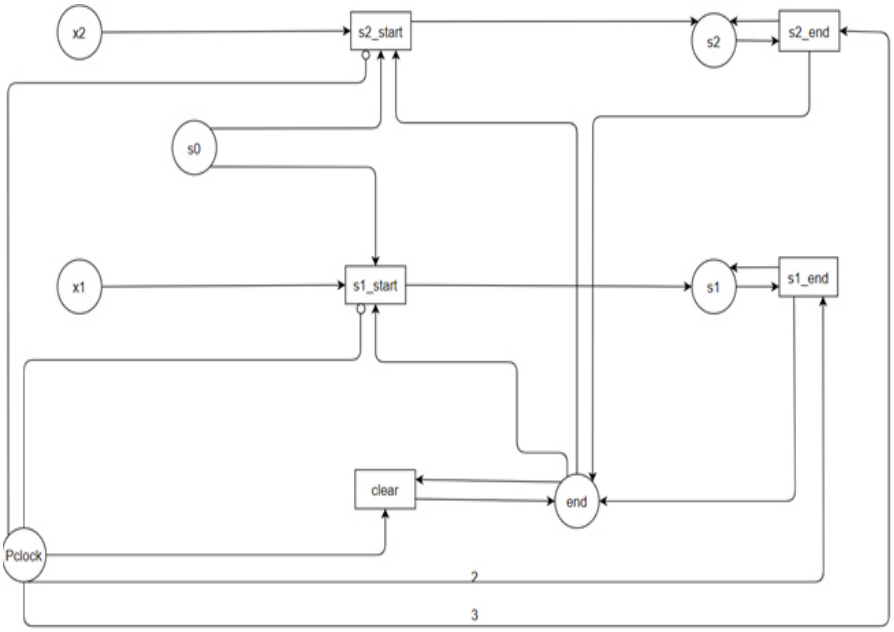

Рисунок 8 – Сеть Петри, эквивалентная временному конечному автомату

На рисунке 9 рассмотрим случай, когда s_1 и s_2 соединены последовательно, а сигнал x_2 – прерывающий.

Здесь по приходу сигнала x_2 мы будем помещать метку в состояние *end* с сохранением исходного условия. Для этого введём переход *s1_terminate*, а также дуги (x_2, *s1_terminate*), (*s1_terminate*, x_2), (*s1_terminate, end*) и (*end, s1_terminate*) – ингибиторная дуга. Теперь, по поступлении сигнала x_2 возможны 2 случая: процесс в *s1* либо завершился, либо всё ещё выполняется. В первом случае сеть переместит фишку из состояния *s1* в *s2*; во втором случае сеть переведёт процесс в состояние end с сохранением исходного сигнала *x2*, и только после очищения счётчика *Pclock* сработает переход *s2_start.*

Обособим случай, когда количество дуг *(pclock, s1_end)*=1, а процесс *s1* завершён по времени. В такой ситуации возможна конкуренция между переходами *clear* и *s1_end.* Во избежание состязательной ситуации добавим ингибиторную дугу (*end, s1_end*) которая будет разрешать конфликт в пользу перехода *clear*:

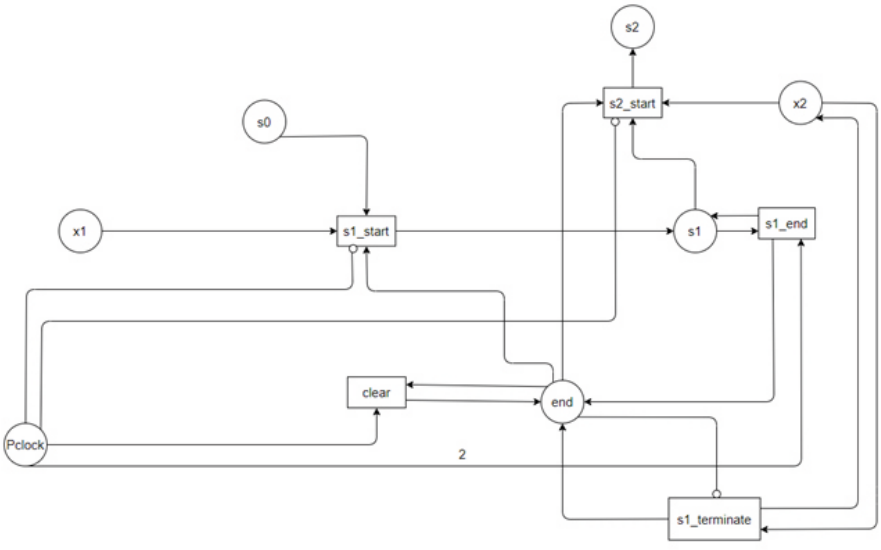

Рисунок 9 – Сеть Петри для автомата, содержащего «прерывающие» переходы

Таким образом, перед конвертированием временного автомата в сеть Петри необходимо разделить множество X на подмножества A и I такие, что $A \cup I = X, A \cap I = \emptyset$. Каждая сеть должна содержать

- $pckock,end,p_0 \in P$ – позиции, общие для всей сети;
- $clear \in T$ – переход для очистки временным меток;
- $(pclock,clear),(clear,end),(end,clear) \in W$ – данная конструкция позволяет очищать скопившиеся временные метки;

Затем, для каждого состояния $s_i \in S$ строим эквивалентные позиции $p_i \in P$, для каждого сигнала $x_i \in X$ так же $p_i \in P$. Для каждого перехода необходимы следующие построения:

- $tstart_t, tend_i \in T$ – эквивалентны состояниям автомата;
- $(p_{i-1}, tstart_t), (x_t, tstart_t), (end, tstart_t), (tstart_t p_t), (p_t, tend_t), (tend_t p_i), (tend_t, end), (pclock, tend_t) * n_i \in W$ – обычные дуги, реализующие логику перехода в следующее состояние;
- $(pclock, tstart_i)$- ингибиторная дуга:

Для каждой дуги, которая прерывает тот или иной процесс, необходимо достроить

- $tterminate_i \in T$;
- $(tterminate_t, x_t), (x_t, tterminate_t), (tterminate\_i, end) \in W$ – обычные дуги;
- $(end, tterminate_i) \in W$ – ингибиторная дуга.

Так же, если для какого либо перехода во временном конечном автомате истинно $\exists t = dt, t \in T$, то в сетях Петри для данного перехода $tend_i \in T$ должна быть построена ингибиторная дуга $(end.tend_t) \in W$.

Заключение. В данной статье было рассмотрено конвертирование временного конечного автомата в сеть Петри с использованием ингибиторных дуг. Формализован метода построения эквивалентной сети Петри по временному конечному автомату.

СПИСОК ЛИТЕРАТУРЫ:

1. Ачасова С. М., Бандман О. Л. Корректность параллельных вычислительных процессов. — Новосибирск: Наука, 1990. — 253 с.

2. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. Моделирование параллельных процессов. Сети Петри. Курс для системных архитекторов, программистов, системных аналитиков, проектировщиков сложных систем управления. - Санкт-Петербург: Профессиональная литература, АйТи-Подготовка, 2014. - 400 с.

3. Веретельникова Е., Теоретическая информатика. Теория сетей Петри и моделирование систем. –Новосибирск: НГТУ, 2018 – 79 с.

4. Котов В. Е. Сети Петри. — М: Наука, 1984. — 160 с.

5. Клейнберг Дж., Тардос Е. Алгоритмы. Разработка и применение. – СПб.: Изд. Питер, 2016. – 800 с.

6. Кормен Томас Х., Лейзерсон Чарльз И., Ривест Рональд Л., Штайн Клиффорд Алгоритмы. Построение и анализ. – М.: Вильямс, 2019. – 1328 с.

7. Питерсон Дж. Теория сетей Петри и моделирование систем. — М: Мир, 1984. — 264 с.

8. Уилсон Р. Введение в теорию графов – М: Вильямс, 2020 г. – 240 с.

9. Gill A. Introduction To The Theory Of Finite State Machines. – New York: McGraw-Hill; First Edition, First Printing edition, 1962. – 207 p.

10. Pashchenko D. et al. The methodology of multicriterial assessment of Petri nets' apparatus //MATEC Web of Conferences. – EDP Sciences, 2016. – Т. 44. – С. 01009.

11. Войнов А. С. и др. Программа преобразования представлений сетей Петри //Новые информационные технологии и системы. – 2015. – С. 29-32.

12. Слепцов А. И., Юрасов А. А. Автоматизация проектирования управляющих систем гибких автоматизированных производств / Б. Н. Малиновский. — Киев: Техника, 1986. — 160 с.

13. Pashchenko D. V. et al. Directly executable formal models of middleware for MANET and Cloud Networking and Computing //Journal of Physics: Conference Series. – IOP Publishing, 2016. – Т. 710. – №. 1. – С. 012024.

14. Скиена С., Алгоритмы. Руководство по разработке. – СПб.: БХВ-Петербург, 2017. – 720 с.

15. Мартяшин Г. В. и др. Возможности применения биоинспирированных алгоритмов в задаче синтеза альтернативных сетей Петри в тензорной методологии исследования СП-структур //Модели, системы, сети в экономике, технике, природе и обществе. – 2017. – №. 1 (21).

16. Захаров Н.Г., Рогов В.Н. Синтез цифровых автоматов: учеб. пособие – Ульяновск:Ульяновский государственный технический университет, 2003. – 135 с.

17. Вашкевич Н. П. Недетерминированные автоматы в проектировании систем параллельной обработки: учеб. пособие / Н. П. Вашкевич. – Пенза:Изд-во Пенз. гос. ун-та, 2004. – 280 с.

18. Pashchenko D. et al. Formal transformation inhibitory safe Petri nets into equivalent not inhibitory //Procedia Computer Science. – 2015. – Т. 49. – С. 99-103.

Статья поступила в редакцию 01.06.2020
Статья принята к публикации 14.09.2020