DOI: 10.15514/ISPRAS–2020–32(6)–12

Automated Analysis of DP-systems Using Timed-Arc Petri Nets via TAPAAL Tool

[1] *A.A. Izmaylov, ORCID: 0000−0003−1721−3347 <zinoviy23@gmail.com>*
[1] *L.W. Dworzanski, ORCID: 0000−0002−0074−7660 <leo@mathtech.ru>*
[1] *National Research University Higher School of Economics,*
Myasnitskaya st., 20, Moscow, 101000, Russia

Abstract. Timed-Arcs Petri nets (TaPN-nets) are a time extension of Petri nets that allows assigning clocks to tokens. System of dynamic points on a metric graph (DP-systems) is another dynamical model that is studied in discrete geometry dynamics and motivated by study of localized Gaussian wave packets scattering on thin structures; as well, DP-systems could be utilized to overapproximate the dynamics of messages scattering in distributed systems. In the latter case, time-temporal properties of DP-systems become a matter of interest. However, there are no tools that enable us to analyse them. In this work, we provide a new approach to automated analysis of DP-systems using the translation of a DP-system into a TaPN-net which is implemented as a TAPAAL plugin. The translation let us use the comprehensive tool support for TaPN-nets (TAPAAL/UPPAAL) to analyze DP-systems dynamical characteristics expressed in TCTL language. We demonstrated how to express some of them and verify time-temporal properties of a DP-system using the suggested approach, and performed experiments to obtain empirical estimates of the tool performance.

Keywords: metric graphs; timed-arc Petri nets; time temporal properties

For citation: Izmaylov A.A., Dworzanski L.W. Analysis of DP-systems Using Timed-Arc Petri Nets via TAPAAL Tool. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 155-166. DOI: 10.15514/ISPRAS-2020-32(6)-12

Acknowledgements. This work was supported by the RFBR grant 20-07-01103 a.

Автоматический анализ дискретных динамических систем на метрических графах с помощью сетей Петри с временными дугами и инструмента TAPAAL

[1] *А.А. Измайлов, ORCID: 0000−0003−1721−3347 <zinoviy23@gmail.com>*
[1] *Л.В. Дворянский, ORCID: 0000−0002−0074−7660 <leo@mathtech.ru>*
[1] *Национальный исследовательский университет «Высшая школа экономики»,*
Россия, 101000, г. Москва, ул. Мясницкая, д. 20

Аннотация. Сети Петри с временными дугами – это временное расширение сетей Петри (TaPN-сети), которое позволяет присваивать таймеры фишкам. Система динамических точек на метрическом графе (DP-система) это другая динамическая модель, которая рассматривается в теории геометрических дискретных динамических систем и, исторически, ее изучение мотивировано изучением распространения локализованных гауссовых волновых пакетов по тонким структурам; кроме того, DP-системы могут использоваться для приближенного представления динамики распространения сообщений в распределенных системах. В этой работе, мы описываем новый подход для автоматического анализа DP-систем используя трансляцию в TaPN сеть, которая реализована как расширение инструмента TAPAAL. Подход позволяет использовать мощные инструменты верификации (TAPAAL/UPPAAL) для анализа динамических характеристик DP-систем, представленных на языке TCTL. В работе продемонстрировано, как можно кодировать временно-темпоральные свойства DP-систем в рамках предложенного подхода, и приведены результаты экспериментальных тестов.

Ключевые слова: метрические графы; сети Петри с временными дугами; темпорально-временные динамические свойства

Для цитирования: Измайлов А.А., Дворянский Л.В. Анализ дискретных динамических систем на метрических графах с помощью сетей Петри с временными дугами и инструмента TAPAAL. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 155-166 (на английском языке). DOI: 10.15514/ISPRAS–2020–32(6)–12

Благодарности: Работа выполнена при поддержке гранта РФФИ 20-07-01103 а.

1. Introduction

The notion of a Petri nets evolved from a chemistry process model to a model of indefinitely expandable computing system consistent with laws of physics in C.A. Petri works [1]. Nowadays, Petri nets are widely-used to model the behaviour of distributed concurrent computer systems and concurrent processes in biology, chemistry, physics, and other fields [1]. Timed-arc Petri nets (TaPN-nets) are an extension of Petri nets with time semantics: tokens are assigned clocks [2]; an inscription on an incoming arc of a transition define tokens of which age can be consumed by the firing of the transition.

Metric graphs are graphs with lengths assigned to edges. A dynamical system consisting of a metric graph and dynamic points moving along the graph edges (DP-system) is a geometrical discrete dynamical system originally motivated by the problem of evolution of wave packets in thin structures. It could be considered as a simplified discrete model of a quantum graph – a metric graph equipped with functions on its edges, a differential operator acting on such functions, and matching conditions on its vertices [3], [4] – with narrow localized wave packets [5], [6]. Quantum graphs occurred as a model or tool in a number of problems in chemistry, physics, engineering, and mathematics since 1930s [7]. It was shown that there exists a correspondence between the statistics of localized solutions on a quantum graph and the dynamics of a DP-system [6]. Points in a DP-system may represent supports of Gaussian wave packets in a quantum graph and/or the projection of wave propagation on medium geodesics. Both models, Petri nets and DP-systems, embrace real-time dynamics of discrete entities moving within a topological structure defined by a graph.

Some results towards the characteristics of the dynamics of DP-systems were recently obtained in [8–11]. The growth of the number of points moving along edges and its asymptotic are studied for metric trees in [10] and, for some special cases, in [6]. In [11], polynomial approximation for the growth of the number of dynamic points moving along metric graphs is studied, and explicit formulae for the first two terms of the polynomial approximation are given. In [6], stabilization of the number of points in a DP-system is studied and explicit formulae for graphs with edges of the same length and star graphs are given, exploiting a connection with analytic number theory problems. While the mentioned quantitative dynamic statistics of DP-systems are studied, finer temporal properties of DP-systems are not; like, for example, which node v_1 or v_2 will receive a point first, or whether e_1 will have received ten or more points by the time e_2 received its first point. In this work, we provide a translation of a DP-system into a TaPN-net [12], which allows reducing the analysis of temporal properties of DP-system to analysis of TaPN-net and conduct it using the effective widely-used TAPAAL tool [13], which is under active development (two golds, a silver, and a bronze medal in MCC'20 [14]).

Temporal properties of Petri nets, such as liveness, boundedness, and reversibility, and their complexities have been extensively studied (see comprehensive review [15]); and, many of them are in PSPACE or, worse, in EXPSPACE complexity classes. The study and development of Petri nets analysis methods is still active; for example, the long-standing question on whether reachability and coverability problems have the same complexity has been recently resolved in [16] showing that reachability is not even elementary, thus, impacting complexity of many problems in other fields, such as formal languages, logic, linear algebra, etc. However, for some restricted Petri net subclasses, many important properties, which are undecidable in general case, are decidable or even have polynomial time complexity.

There is a lot of ways to introduce time constraints in Petri nets [17-18]. Unfortunately, it was shown that almost any of semantical extensions makes Petri nets Turing-complete and, by Rice-Uspensky theorem, many of general behavioural problems immediately become undecidable. The widely known time extensions of Petri nets – Time Petri nets [19] and Timed (Duration) Petri nets [20] – are Turing-complete as they admit urgency and allow to model unbounded counters. Time extensions of Petri nets, as well as other real-time models, are under active study as, for many real-world software/hardware systems, time related aspects like performance, time-outs, delays, and latency are crucial for correct functioning [21-23]. A time semantics with restricted urgency was recently suggested for TaPN-nets in [24]; the suggested semantics allows urgent transitions to consume tokens only from the bounded places of a Petri net, and this restriction makes some behavioural problems decidable for TaPN-nets. In [25], author suggested an approach to use timing specifications to improve the behavioural properties of an untimed P/T-net with an example of making live and unbounded untimed P/T-net live and bounded by cutting off token generators using time-based constraints.

TaPN-nets attract our attention as they feature dynamics that makes it possible to model DP-systems. In [26], timed-arc Petri nets were consistently combined with ‘nets-within-nets’ hierarchical structure and sound time semantics was provided; also, the decidability of behavioural coverability-related properties using the notion of well-structured transition systems was established. For recent review of results on TAPN-nets and their verification, we refer to [27]. Moreover, TaPN-nets have comprehensive tool support via TAPAAL and UPPAAL software. Tool UPPAAL makes it possible to model and verify networks of timed automata. TAPAAL is a tool for modelling, simulation, and verification of TaPN-nets; it can utilize UPPAAL as a verification engine.

While a lot of results on temporal properties of timed and untimed versions of Petri nets were obtained, quantitative statistics of the behaviour of Petri nets and their extensions are studied to much lesser extent. The translation of TaPN-nets into DP-systems can be used to overapproximate the number of different age-values in the TAPN-net. This translation is a part of the tool implemented within the frame of this works – it extracts a metric graph in GraphML format from a TaPN-net allowing one to use existing software for metric graphs to overapproximate the stabilization time or growth rate of the number of clocks in the TaPN-net.

In Section 2, basic notions and notations are given. Section 3 covers the translation between DP-systems and TaPN-nets and implementation details. In Section 4, we demonstrate how to verify time-temporal properties of a DP-system using the suggested approach; results of performance experiments are provided. Section 5 concludes the paper with some discussion.

2. Preliminaries

By $\mathrm{N}, \mathrm{Q}_{\geq 0}, \mathrm{R}$, we denote the sets of natural, non-negative rational, and real numbers, respectively. The set of open and closed intervals over $\mathrm{Q}_{\geq 0} \cup \{\infty\}$ is denoted by $\mathrm{I}\,(\mathrm{Q}_{\geq 0})$. For a set S, a *bag* (*multiset*) m over S is a mapping $m : S \to \mathrm{N}$. The set of all bags over S is denoted by N^S. We denote addition and subtraction of two bags by $+$ and $-$, the number of all elements in m taking into account the multiplicity by $|m|$, and comparisons of bags by $=, <, >, \leq, \geq$. We start by giving the definition of a metric graph [3].

Definition 1 (Metric graph). *A graph Γ is said to be a metric graph, if*

- *each arc a is assigned a positive length $l(a) \in (0, \infty)$. An arc with $l(a) = \infty$ has only one incident vertex and is called a lead;*
- *the lengths of the arcs that are reversals of each other are assumed to be equal;*
- *a coordinate $x(a) \in [0, l(a)]$ increasing in the direction of the arc is assigned on each arc;*

- *the relation $x(a') = l(a) - x(a)$ holds between the coordinates on mutually reversed arcs.*
- *for arc a, the length $l(e)$ of its support edge e is equal to $l(a)$.*

A state s of a DP-system D on a metric graph Γ is a set of points located on the arcs of Γ. When time starts to flow, each point p moves along its arc a direction from 0 to $l(a)$ with the same velocity; its position is denoted by $x(p)$. If p reaches a vertex v, new points are issued to all the outgoing arcs of v (intuitively, this corresponds to wave packet scattering). New points generation may result in more than one point on some arcs. Each produced point p' starts moving along corresponding e. When more than one points reach a vertex simultaneously at t, on each outgoing arc, only one point is produced, as if only one point has reached the vertex at t; i.e., points met on a vertex fuse, and each coordinate of an arc can carry only one dynamic point. However, points do not collide anywhere on edges except vertices, i.e., if two points met on an edge, they both continue their movement towards their own directions. In fig. 1, the initial set of points consists of two points in vertices v_1 and v_3. The point in v_1 produces a new point on edge $\{v_1, v_2\}$, The point in v_3 produces points on edges leading to vertices v_2, v_4, v_5, v_6, v_7. After a time unit, there are no points in v_1 and v_3 (coloured gray), but there are points (coloured black) moving from v_2 and v_3 to their adjacent vertices.

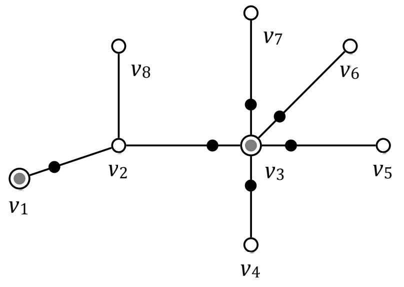

Fig.1. System of dynamic points D on metric graph Γ

The stabilization time $t_s(\Gamma)$ of a metric graph Γ with commensurable edge lengths is the value of the period of time from the starting point to the point in time when the number of points $N_\Gamma(t)$ on the graph has been stabilized [6], *i.e.*, $\forall t \geq t_s(\Gamma) : N_\Gamma(t) = N_\Gamma(t_s(\Gamma))$. Henceforth, when stabilization time is discussed, discrete time instants when points collapse at vertices are excluded from consideration as, strictly, at these instants the number of points can decrease.

A place/transition net (PT-net) is a Petri net with indistinguishable tokens.

Definition 2 (Place/transition nets). A PT-net is a tuple $< P, T, F, \gamma >$, where

- *P and T are disjoint finite sets of places, respectively, transitions;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);*
- *$\gamma : F \to N$ is a weight function;*

For an element $x \in P \cup T$, an arc $< y, x >$ is called an *input arc*, and arc $< x, y >$ – an *output arc*. A *preset* $\bullet\, x$ and a *postset* $x\, \bullet$ are subsets of $P \cup T$ such that $\bullet\, x = \{y \mid < y, x > \in F\}$ and $x\, \bullet = \{y | < x, y > \in F\}$. A *marking* of N is a function $m: P \to \mathrm{N}$. A pair $< N, m >$ of a PT-net and a marking is called a marked net.

Let $N = < P, T, F, \gamma >$ be a PT-net. A transition $t \in T$ is *enabled* in a marking m iff $\forall p \in \bullet\, t \Rightarrow m(p) \geq \gamma < p, t >$. An enabled transition t can *fire* yielding a new marking $m'(p) = m(p) - \gamma < p, t > + \gamma < t, p >$ for each $p \in P$ (denoted $m \to^t m'$).

Now, we provide the definition of Timed-Arc Petri nets with token-based time semantics and urgency [2], [12], [24].

Definition 3 (Timed-Arc Petri net with Urgency). A TaPN-net is a tuple $TaPN = < N, \gamma^t, U >$, where

- $N = < P, T, F, \gamma >$ *is a PT-net called the skeleton of TaPN and denoted by* $S(TaPN)$*;*
- $\gamma^t: P \times T \to I(Q \geq 0)$ *is a set of token-age constraints on arcs;*
- $U : T \to Q_{\geq 0}$ *is a set of urgency constraints on transitions.*

The marking $m = < m_s, m_t, m_u >$ of a TaPN-net $TaPN$ consists of a marking m_s of $S(TaPN)$, a time marking $m_t : Tok(m_s) \to Q_{\geq 0}$ that assigns clocks to tokens, and an urgency marking $m_u : T(m_s) \to Q_{\geq 0}$ that assigns clocks to transitions, where $T(m_s)$ comprises all transitions and $Tok(m_s)$ comprises all tokens of the marked PT-net $< S(TaPN), m_s >$. The urgency constraint $U(t)$ means that t must fire if t has been enabled for $U(t)$ units of time. The token-age constraint $\gamma^t(p,t)$ means that t may fire only by consuming a token z in p with $m_t(z) \in \gamma^t(p,t)$. The urgency U of a transition is depicted as a number near the transition. The time constraints γ^t of an arc are depicted as an interval on the arc.

The operational semantics of TaPN-nets is defined by incorporating time constraints into the firing rules of PT-nets. A transition t is enabled in the marking $m = < m_s, m_t, m_u >$, if t is enabled in $< S(TaPN), m_s >$ and time constraints of t are satisfied, *i.e.*, each token z from a place p involved in the firing of t satisfies $m_t(z) \in \gamma^t(p,t)$.

A *time elapsing* step corresponds to the elapsing δ time units in each clock of the marking m. We assume that all token clocks and transition clocks run at the same pace. We denote by $m + \delta$ the marking with all its clocks increased by δ, *i.e.*, for each token $z \in m_s : (m_t + \delta)(z) = m_t(z) + \delta$, and for each transition $t : (m_u + \delta)(t) = m_u(t) + \delta$. Under urgency restrictions, a time elapsing step δ is allowed if there are no $\delta' \in [0, \delta)$ such that the $m + \delta'$ marking has urgent transitions.

3. Automated translation of DP-systems into TaPN-nets

In the context of communication networks and computer systems, we may consider a distributed system of communicating reactive sensor-nodes (or other computational devices). One of the nodes is a server that initiates communication by sending control messages (signals) while other sensor-nodes react on such signals. On the assumption that the processing speed of nodes is much higher than the speed of signal propagation, when a node receives one or more signals from its neighbour nodes, it responds instantly by sending signals to some of its neighbours. In an extreme case, each node sends signals to all of its neighbours upon each message arrival. This extreme behaviour corresponds to the dynamics of a DP-system enabling us to use metric graphs to overapproximate messages scattering in networks.

In the latter case, time-temporal properties of DP-systems become a matter of interest [28]. However, up to the knowledge of authors, there are no tools that enable us to conduct analysis of such natural time-temporal properties as:

- Is it possible that more than N signals come to node X in K seconds;
- Is it possible that in the next K minutes two messages come with the difference of their time moments less that ε milliseconds;
- For the system initial phase of duration N, if a point comes to node X at time T, then no points comes to node Y within K seconds from T;

In this section, we provide an algorithm for the translation of DP-systems in TaPN-nets. Tool TAPAAL supports verification of specifications in timed computation tree logic (TCTL) language, which allows expressing time and temporal properties of process dynamics [29] [30]. In the next section, we demonstrate how to express these properties in TCTL-properties of converted TaPN-nets.

To simulate *DP*-systems with models of Petri nets with real-time semantics, we need to represent points moving along edges using clocks in a Petri net. The advance of a point along an edge in a metric graph is modelled with the progress of the corresponding clock in a Petri net.

The number of points on a metric graph is not fixed and may grow (infinitely when edge lengths are incommensurable); therefore, it is not possible to model *DP*-system using clocks in time or timed Petri nets [19], [20] as these models have finite structurally-determined number of clocks. In *TaPN*-nets, clocks are assigned to tokens [12], and the number of clocks can grow along with the number of tokens in the net.

We start with the description of *DP*-system to *TaPN*-net translation procedure, and, later, cover some design and technical details on the implementation of the translation.

We denote the set of points on the edge e by $P(e)$. For a lead $< v_1, v_2 >$, one of v_1 and v_2 is marked with infinity. Note that, as movement of a point along an edge corresponds to a continuous process, we model edges of DP-system with TaPN-net places; and, as arrival of a point at a vertex corresponds to a discrete event, we model vertices of DP-system with TaPN-net transitions.

3.1 Algorithm for translation from a DP-system to a TAPN-net procedure

Step 1. For each edge e connecting vertices a and b in D, we create places e_{ab} and e_{ba} in N_{TA};

Step 2. For each lead $< a, b >$ in D with b marked with infinity, an arc from vertex a to a distinct infinity-vertex is introduced in N_{TA}. We need to handle points moving away from vertex a separately.

(a) create place e_{ab}, transition t_{ab} with urgency 0, and an arc from e_{ab} to t_{ab} with time interval [0,0];

(b) for each point on the arc $< a, b >$

  i. create a token with 0 time at place e_{ab};

  ii. create transition t_{ba} with urgency 0;

(c) for each point p_i on arc $< b, a >$

  i. create place e^i_{ba} with a token;

  ii. create an arc from e^i_{ba} to t_{ba} with time interval $[x(p_i), x(p_i)]$;

(d) for each arc from a to c, where $c \neq b$, create an arc from t_{ba} to e_{ac};

Step 3. For each already created place e_{ab} in N_{TA} such that vertex a is not marked with infinity in D

(a) for each point p on arc $< a, b >$

  i. create token at e_{ab} with timer set to $x(p)$;

(b) create transition t_{ab} with 0 urgency if b is not marked with infinity;

(c) create an arc with time interval $[l(e), l(e)]$ from e_{ab} to t_{ab};

(d) for each arc $< b, c >$ in D, create an arc from t_{ab} to e_{bc} in N_{TA};

(e) create transition t'_{ab}, an arc from e_{ab} to t'_{ab} with multiplicity 2, urgency 0, and zero time interval, and arc from t'_{ab} back to e_{ab}.

An example of such conversion is illustrated in fig. 2. As each undirected edge in D corresponds to two opposite arcs, the number of places in NTA on the right is twice as the number of vertices in D on the left. Transitions corresponds to the event of a point reaching a vertex; auxiliary transitions are introduced to clean redundant tokens from edge-places in NTA to handle events when multiple point came to a vertex simultaneously.

3.2 Algorithm for translation from timed-arc Petri nets to metric graphs

To use the results on the asympotics and estimates on growth and stabilization time of the number of dynamic points in *DP*-system [6] to overapproximate *TaPN*-net the number of different token age-values, we implement the translation of a restricted class of *TaPN*-nets into *DP*-systems.

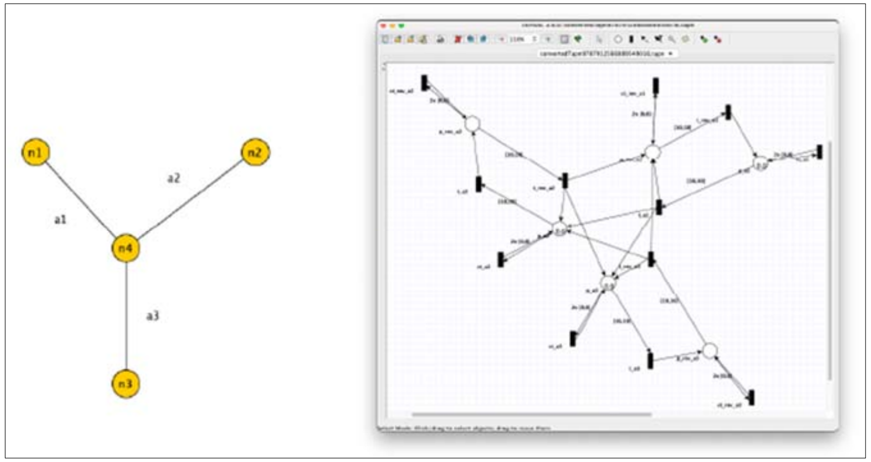

Fig.2. Metric graph D on the left and the resulting TAPN N_{TA} on the right

Let us consider a class of TaPN-nets under the following restrictions:

1) The skeleton of *TaPN*-net is a marked graph, i.e., every place has exactly one incoming arc, and exactly one outgoing arc [1];
2) All urgency time specifications are 0;
3) All arcs are labelled with point-intervals only, i.e., intervals of form $[t, t]$.

Let NTA be an input TaPN-net and D be the resulting DP-system. The next procedure converts the TaPN-net to a DP-system. The DP-system D generates point each time NTA produces a new token with timer.

1) For each transition t_i in N_{TA}, create vertex v_i in D;
2) For each transition t_i, place p, and transition t_j, such that there is an arc from t_i to p and an arc from p to t_j
   a) create an edge connecting vertices v_i and v_j (which correspond to transitions t_i and t_j, respectively) with a length equal to the time interval on the arc from p to t_i; this procedure may introduce multiple edges or self-loops, thus, an intermediate graph may be not a metric graph; we provide a resolving strategy after the procedure;
   b) for each token in p, put a point on beginning of edge from v_i to v_j, *i.e.,* at vertex v_i;
3) For each place p, which has only inbound arcs or only outbound arcs
   a) for each adjacent transition t, create a lead from corresponding vertex v;
   b) if p has only outbound arcs, then for each token in p, for each adjacent transition t_i, put a point on the lead incident to corresponding vertex v_i moving towards v_i and located on the distance equal to the point-interval on arc $< p, t_i >$.

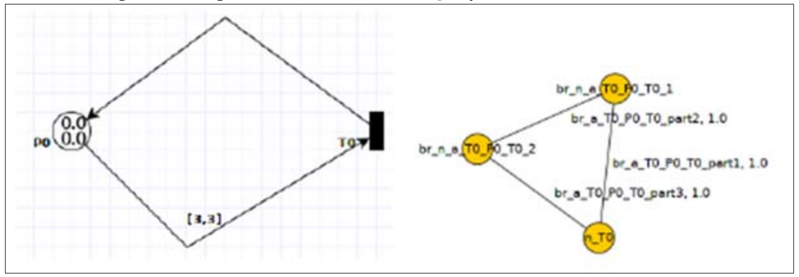

Fig.3. Example of self loop break

To resolve multiple edges and self-loops introduced during the translation, we break one of edges e into two edges by introduce a new vertex; lengths of these new edges are equal to the half $l(e)$. For self loops, we need to introduce two vertices, and they breaks edge e into three parts with lengths equal to a third of $l(e)$. This step, firstly, breaks the loop with one edge, and, secondly, breaks multiple edges; the result of such step for a loop transition is illustrated in fig. 3.

3.3 Implementation details

The translation is implemented in tool **mg-to-tapn** as a TAPAAL extension; the tool uses TAPAAL internal representation for storing and analysis of TaPN-nets. A JSON-based file format for processing DP-systems was developed and its support was implemented. The file format is an extension of JSON Graph Format [31]. To allow visualization of DP-systems, a support for GraphML-based format compatible with yEd Graph Editor [32] was implemented.

As **mg-to-tapn** is an extension of TAPAAL, the following functions can be utilized both through command line interface or graphical user interface of TAPAAL.

- Translate DP-system in JSON format to TaPN-net in TAPAAL representation;
- Translate TaPN-net in TAPAAL representation to DP-system in JSON format;
- Translate TaPN-net in TAPAAL representation to DP-system in GraphML-based yEd-compatible format.

4. Checking TCTL properties of DP-system and experimental results

In this section, we demonstrate how to check DP-system time-temporal properties of a metric graph using the implemented translation and provide results of performance experiments.

4.1 Checking time-temporal properties of a metric graph

Tool *TAPAAL* does not directly support the whole class of *TCTL* properties; therefore, for time-temporal properties, we need to combine supported verification engines with additional auxiliary net fragments to capture such properties. We demonstrate how to encode the following propertiesusing this technique.

i. **Is it possible that more than N signals come to node X in K seconds**. Firstly, we need to capture the property that more than N signals come to node X. Arrival of a signal to X from Y along arc $< Y, X >$ corresponds to a firing of transition t_{YX} in *TaPN* (see naming strategy in section 3). We add a place p'_X, which will be used as a counter for the number of signals-tokens that came to X. Then, we connect each t_{YX} to p'_X with an arc; each firing of t_{YX} will produce a token in p'_X. We call such a place *counter for* X. In addition, we need to determine a time point when K seconds has passed. We introduce a new place p_{start} and put a token in p_{start}. We add urgent transition t_{time} and connect p_{start} and $t_{time}$with an arc specified by time interval $[K, K]$. So, place p_{start} becomes empty when K seconds has passed. We call such a net fragment consisting of place p_{start} with a token, transition t_{time}, and arc $< p_{start}, t_{time} >$ specified by $[K, K]$ as *timer for* K.

   Finally, we can express the property as whether there is a reachable state when p'_X has K or more tokens, and p_{start} has exactly one token. This property is purely temporal while time features are captured using time specifications in *timer for K*. In CTL, the property is expressed as $EF((m(p'_X) \geq N) \& (m(p_{start}) = 1))$.

ii. **Is it possible that in the next K minutes two messages come with the difference of their time moments less that ε milliseconds**. To check the property, we add *timer for K* construction, *counter for X*, and transition t'. We add arc $< p'_X, t' >$ labelled with $[\varepsilon, \varepsilon]$. To make , t' urgent, we use age invariant $Inv: \leq \varepsilon$ on p'_X. If there are two or more tokens in p'_X, , then those tokens come to node X with time difference less than ε. The corresponding CTL property is $EF((p'_X \geq 2) \& (p_{start} = 1))$.

iii. **For the system initial phase of duration N, if a point comes to node X at time T, then no points comes to node Y within K seconds from T.** We add *counter for* X, *counter for* Y, and *timer for* N. We add transition t'_X, arc $< p'_X, t'_X >$labelled with $[K, K]$, age invariant $Inv: \leq K'$ on p'_X making t'_X urgent, urgent transition t'_Y, and arc $< p'_Y , t'_Y >$. If a token is in p'_X for K seconds, then t'_X fires consuming it; if a token came to p'_Y, it leaves the system via t'_Y immediately.
The property become violated if p'_X and p'_Y hold tokens simultaneously, as it corresponds to a state where a token, which timer value is less than K, is in X, and a token came to Y . The resulting CTL property is $AG\neg((p'_X \geq 1) \& (p'_Y \geq 1) \& (p_{start} = 1))$.

Table 1. Performance results for verifying the first property

Graph type/ size (in nodes)	**3**	**5**	**10**	**20**	**30**
Complete	0.05s	0.11s	1.36s (11mb)	41.29s (26mb)	–
Key	0.03s	0.07s	0.09s	0.4s (6mb)	11.91s (86mb)
Cycle	0.02s	0.03s	0.03s	0.02s	0.03s
Star	0.05s	0.09s	1.10s (13mb)	51.53s (353mb)	> 300s (> 1.8gb)

Table 2. Performance results for verifying the third property

Graph type/ size (in nodes)	**3**	**5**	**10**	**20**	**30**
Complete	0.01s	0.01s	0.03s	0.65s (11mb)	–
Key	0.01s	0.01s	0.08s	4.21s (45mb)	> 300s (> 1.8gb)
Cycle	0.01s	0.01s	0.01s	0.01s	0.06s / 2mb
Star	0.01s	0.01s	> 300s (> 1.6gb)	14.16s (120mb)	> 300s (> 1.1gb)

4.2 Experimental results

We conducted experiments on metric graphs of different topology: star graph, complete graph, cycle graph, and «key» graph (linear graph joined with a cycle of 3 elements on the end). The metric graphs were converted to *TaPN*-nets in *TAPAAL* format. The first and third property, encoded as suggested in the previous section, were checked for the converted metric graphs; verification time and memory consumed were measured.
The results for the first property are given in Table 1. For this experiment, we checked the first property for the initial fragment of 1000 time units. Time and consumed memory were measured; if the amount of consumed memory was negligible, the value is not provided. The results for metric graph K_{30} were not obtained as more than a thousand nodes in the resulting *TaPN*-net exceeded some internal restrictions of *TAPAAL*. The low verification time for a cycle metric graph is stipulated by the low node degrees in the graph thus leading to a smaller number of transitions in the converted *TaPN*-net. The results for the third property are given in Table 2. For this experiment, we checked the second property with $K = 20$ and $N = 1000$. We see almost the same picture with the exception of that for key metric graphs the memory restrictions come in play earlier. The experiments were performed on a personal computer with 2 GHz Quad-Core Intel Core i5 processor and 16 GB of memory (3733 MHz LPDDR4X) running under macOS 11.0.1 operating system.

5. Conclusion

In this paper, we developed an approach that enable us to analyze TCTL-properties of DP-systems based on the implementation of the translation between DP-systems and TaPN-nets. We demonstrated how to conduct analysis of TCTL properties of DP-systems by adding an auxiliary net

fragment. For several time-temporal properties of metric graphs, we showed how to express the properties in TAPAAL using an auxiliary subnet and TAPAAL TCTL queries.
The automated translation allows using the well-known efficient tool TAPAAL to carry out simulation and verification (analysis of TCTL properties) of DP-system dynamics and interpreting results for DP-systems on a given restricted subclass of TaPN-nets.
The experiments showed that for scarce mid-sized graphs analysis could be done quite effectively. As these experiments were conducted for the first time, to get more straightforward results, we neither used any optimizations nor tried to encode properties in a more compressed manner. Therefore, we expect that optimizations could considerably improve performance on the benchmark tests as complete and star graphs possess much symmetry.
To improve the precision of the overapproximation of a TaPN-net with a DP-system, the already known counting functions and asymptotics for undirected metric graphs shall be extended to directed metric graphs. This direction is being under current research.

References / Список литературы

[1]. Reisig W. Understanding Petri Nets – Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013, 257 p.
[2]. Hanisch H.-M. Analysis of place/transition nets with timed arcs and its application to batch process control. Lecture Notes in Computer Science, vol. 691, 1993, pp. 282–299.
[3]. Berkolaiko G. and Kuchment P. Introduction to quantum graphs. American Mathematical Society, 2013, 270 p
[4]. Berkolaiko G., Carlson R., Fulling S. A., Kuchment P., eds. Quantum Graphs and Their Applications. Proc. of an AMS-IMS-SIAM Joint Summer Research Conference on Quantum Graphs and Their Applications. American Mathematical Society, 2006, 307 p.
[5]. Чернышев В.Л. Нестационарное уравнение Шрёдингера:статистика распространения гауссовых пакетов на геометрическом графе. Труды Математического института им. В.А. Стеклова, том 270, 2010 г., стр. 249–265 / Chernyshev V. L. Time-dependent Schrödinger equation: Statistics of the distribution of Gaussian packets on a metric graph. Proceedings of the Steklov Institute of Mathematics, vol. 270, 2010, pp. 246–262.
[6]. Tolchennikov A.A., Chernyshev V.L., Shafarevich A.I. Behavior of quasi-particles on hybrid spaces. relations to the geometry of geodesics and to the problems of analytic number theory. Regular and Chaotic Dynamics, vol. 21, no. 5, 2016, pp. 531–537.
[7]. Exner P. and Lipovsky J. Topological bulk-edge effects in quantum graph transport. Physics Letters A, vol. 384, issue 18, 2020, article id 126390.
[8]. Толченников А.А., Чернышев В.Л., Шафаревич А.И. Асимптотические свойства и классические динамические системы в квантовых задачах на сингулярных пространствах. Нелинейная динамика, том 6, no. 3, 2010 г, стр. 623-638 A. A. Tolchennikov, V. L. Chernyshev, A. I. Shafarevich, Asymptotic properties and classical dynamical systems in quantum problems on singular spaces, Russian Journal of Nonlinear Dynamics, vol. 6, no. 3, 2010, pp. 623–638 (in Russian).
[9]. Chernyshev V.L., Tolchennikov A.A. Asymptotic estimate for the counting problems corresponding to the dynamical system on some decorated graphs. Ergodic Theory and Dynamical Systems, vol. 38, no. 5, 2018, pp. 1697–1708.
[10]. Chernyshev V.L., Tolchennikov A.A. Correction to the leading term of asymptotics in the problem of counting the number of points moving on a metric tree. Russian Journal of Mathematical Physics, vol. 24, no. 3, 2017, pp. 290–298.
[11]. Chernyshev V.L., Tolchennikov A.A. The second term in the asymptotics for the number of points moving along a metric graph. Regular and Chaotic Dynamics, vol. 22, no. 8, 2017, pp. 937–948.
[12]. Bolognesi T., Lucidi F., and Trigila S. From timed Petri nets to timed LOTOS. In Proc. of the IFIP WG6. 1 Tenth International Symposium on Protocol Specification, Testing and Verification X, 1990, pp. 395–408.
[13]. David A., Jacobsen L., Jacobsen M., Jørgensen K., Møller M., and Srba J. TAPAAL 2.0: integrated development environment for timed-arc Petri nets. Lecture Notes in Computer Science, vol. 7214, 2012, pp. 492–497.
[14]. Model checking contest 2020. Available at: https://mcc.lip6.fr/models.php, accessed June 2020.

[15]. Esparza J. Decidability and complexity of Petri net problems – an introduction. Lecture Notes in Computer Science, vol. 1491, 1998, pp. 374–428.

[16]. Czerwinski W., Lasota S., Lazic R., Leroux J., Mazowiecki F. The reachability problem for Petri nets is not elementary. In Proc. of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019, pp. 24–33.

[17]. Berard B., Cassez F., Haddad S., Lime D., and Roux O. H. Comparison of different semantics for time Petri nets. Lecture Notes in Computer Science, vol. 3703, 2005, pp. 293–307.

[18]. Brown C. and Gurr D. Timing Petri nets categorically. Lecture Notes in Computer Science, vol. 623, 1992, pp. 571–582.

[19]. Merlin P. A study of the recoverability of computer systems. Ph. D. Thesis, Dept. of Information and Computer Science, University of California, Irvine, CA, 1974.

[20]. Ramchandani C. Analysis of asynchronous concurrent systems by timed Petri nets. Ph. D. Thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, 1974.

[21]. Tvardovskii A.S., Yevtushenko N.V. On reduced forms of initialized Finite State Machines with timeouts. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 2, 2020, pp. 125-134. DOI: 10.15514/ISPRAS-2020-32(2)-10.

[22]. Твардовский А.С., Лапутенко А.В. О возможностях автоматного описания параллельной композиции временных автоматов. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 25-40 / Tvardovskii A.S., Laputenko A.V. On the possibilities of FSM description of parallel composition of timed Finite State Machines. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 25-40 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-2.

[23]. Vinarskii E.M., Zakharov V.A. On the verification of strictly deterministic behaviour of Timed Finite State Machines. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 3, 2018, pp. 325-340. DOI: 10.15514/ISPRAS-2018-30(3)-22.

[24]. Akshay S., Genest B., and Helouet L. Decidable Classes of Unbounded Petri Nets with Time and Urgency. In Proc. of the 37th International Conference on Application and Theory of Petri Nets and Concurrency, 2016, pp. 301–322.

[25]. Dworzanski L.W. Bounding untimed Petri net using timing operation. Report on doctoral intermediate thesis results, 2013-09-16. Available at http://mathtech.ru/dworzanski/talks/bounding.html

[26]. Dworzanski L.W. Consistent timed semantics for nested Petri nets with restricted urgency. Lecture Notes in Computer Science, vol. 9884, 2016, pp. 3-18.

[27]. Jacobsen L., Jacobsen M., Møller M. H., Srba J. Verification of timed-arc Petri nets. Lecture Notes in Computer Science, vol.6543, 2011, pp. 46-72.

[28]. Shoshmina I.V. Developing formal temporal requirements to distributed program systems. System informatics, no. 8, 2016, pp. 21-32.

[29]. Alur R, Courcoubetis C, Dill D. Model-checking in dense real-time. Information and Computation, vol. 104, no. 1, 1993, pp. 2-34.

[30]. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. БХВ-Петербург, 2010 г., 552 стр. / Karpov Yu. G., Model checking. Verification of parallel and distributed systems. BHV-Peterburg, 2010, 552 p. (in Russian).

[31]. JSON Graph Format (JGF). Available at: http://jsongraphformat.info, accessed November 2020.

[32]. yEd Graph Editor: High-quality diagrams made easy. Available at: https://www.yworks.com/products/yed, accessed November 2020.

Information about authors / Информация об авторах

Леонид Владимирович ДВОРЯНСКИЙ– доктор естественных наук (Doctor rerum naturalium), независимый исследователь. Сфера научных интересов: методы анализа динамических свойств дискретных динамических систем, моделей параллельных и распределенных систем, моделей систем реального времени, вполне структурированных систем переходов.

Leonid Wladimirovich DWORZANSKI – doctor of natural sciences (Doctor rerum naturalium), independent researcher. Research interests: analysis methods for discrete event dynamical systems, models for parallel and distributed computation, models for real-time systems, well-structured transition systems.

Александр Александрович ИЗМАЙЛОВ проходит обучение в департаменте программной инженерии факультет компьютерных наук НИУ ВШЭ.

Aleksandr Aleksandrovich IZMAYLOV studies at School of Software Engineering Faculty of Computer Science National Research University Higher School of Economics.