Modular Construction of Time Petri Net Reachability Graphs

Ilona Knizhnikova
National Research University Higher School of Economics
iknizhnikova@gmail.com

Leonid Dworzanski
National Research University Higher School of Economics
leo@mathtech.ru

***Abstract*—Time Petri nets are an extension of Petri nets formalism with time specifications on transitions. The formalism is convenient for model distributed systems and enables capturing the time characteristics of distributed system activities. The primary tool for models behaviour understanding is reachability graph. In [10] the algorithm for constructing Time Petri net reachability graph was suggested. It is based on *essential* states, but the number of states in the resultant net reachability graph increases when time specification are scaled up, while the behaviour of the net is invariant under time specification scaling. We study the modification of this algorithm that allows to build Time Petri nets reachability graphs more efficiently using common divisors of the time specification in the components of a Time Petri net.**

***Keywords*—time petri nets, reachability graph, essential states.**

I. Introduction

Petri nets are a popular formalism for modelling concurrent systems. Different extensions of Petri nets and their applications are extensively studied in the literature [12], [3], [2], [11], [4], [6]. Obviously, time is very important aspect of systems behaviour. Time restrictions like "this action can take from N to M seconds" are crucial for real-time system, net protocols, control systems et cetera. Time can be introduced into the Petri nets formalism in many different ways [1]. Moreover, even timeless distributed systems are hard to understand [8], [5]. There are two popular types of such nets: time Petri nets and timed Petri nets. Both of these modifications can simulate counter machines, i.e. are Turing-complete. Both can be used to model systems with time specifications, but in this article only time Petri nets are considered.

Time restrictions make behaviour of Time Petri nets models extremely hard to understand. It means that we have no options but to use computer aided means to check the correctness of a developing system or to analyze already constructed one. The crucial tool to understand the behaviour of a model is reachability graph. The problem of time Petri nets model checking can be solved via essential-states-based algorithm for constructing reduced reachability graphs [9]. But when all time specifications of a Time Petri net are multiplied by a constant, the size of reachability graph is increased or decreased while the behaviour of the net has not changed. In this work we study how to use this property of the algorithm to reduce the space requirements for analysis of a Time Petri net.

The paper is organized as follows. To start with, we provide basic notations of Petri nets. Then we define the Time Petri nets formalism. Then we provide a Time Petri net example which captures the process of a university course from the student viewpoint. After that we apply our modification to build the reachability graphs of the net components of the example and provide the obtained results. The paper ends with conclusion.

II. Preliminaries

Petri net is a well known formalism widely used to model concurrent systems. Petri nets offer graphical notation and rigorous formal semantics. A Petri net is a marked directed bipartite graph, where the structure of the graph defines the behaviour of the model while the marking of the graph defines the current state.

Definition 2.1: A Petri (P/T-net) net is a 4-tuple (P, T, F, W) where

- P and T are disjoint finite sets of places and transitions, respectively;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs;
- $W : F \to \mathbb{N}$ — an arc multiplicity function, that is a function which assigns every arc a positive integer number (arc multiplicity).

Following extension of W we denote as $\widetilde{W}$:

$$\widetilde{W}(x,y) = \begin{cases} n, & xFy \wedge W(x,y) = n \\ 0, & \neg xFy \end{cases}$$

A marking of a Petri net (P, T, F, W) is a multiset over P, i.e. a mapping $M : P \to \mathbb{N}$. By $\mathcal{M}(N)$ we denote a set of all markings of a P/T-net N. We say that the transition t in a P/T-net $N = (P, T, F, W)$ is active in the marking M iff for every $p \in \{p|(p,t) \in F\} : M(p) \geq \widetilde{W}(p,t)$. An active transition may fire, resulting in a marking M' such that $\forall p \in P : M'(p) = M(p) - \widetilde{W}(p,t) + \widetilde{W}(t,p)$.

Another notion we will use is hammock. While it is well known notion we will recall it informally. Hammock is a part of the graph such, that only two vertexes of hammock are linked with the rest of the graph. These vertexes are called starting and ending vertexes of hammock. There can be the subgraph of arbitrary complexity between these two vertexes, but this system should have no link to the graph except starting and/or ending vertex of hammock. [7]

III. Motivating example

In this section, we provide the example of a time Petri net $Tnet$ that models the flow of some course.

This work is licensed under the Creative Commons Attribution License.

The first transition of the $Tnet$ denotes the beginning of a semester and the beginnig of a course. The process consists of two almost independent scenarios, each of whose is represented by a separate net hammock. First of them (starting with the place *start1* and ending with *end1*) can be interpreted as the process of preparing for exams and working with a teacher. Second one models an examination process. The course can be completed iff these parts are performed successfully, i.e. final mark depends on both of them. But they do not block each other — a student has to pass the exam without regard to his work during the semester.

The process starts with a transition *course started* firing, which adds tokens to the places *start1* and *start2*. Then two hammocks are performing independently.

We start with considering the upper hammock. Firstly, transition *start solve problems* fires. It represents the beginning of student's work and adds tokens to the places *student* and *no more problems*. When both of these places contain tokens — student isn't solving a problem and is ready for actions. The course can be ended through firing the transition *all problems solved*, or the student can get some new problems to solve (transition *hand-in problem*). In the second case, the student needs to solve the problem. He or she starts with meeting with a tutor (going to place *meeting with tutor*), discusses the problem (transition *discussion with tutor* moves token to the place *student* and this enables transition *start work*) and then student starts working on it, and comes to some decision. (Transition *start work* fires and moves token to the place *solving*). After that, the problem is technically solved and transition *problem solved* fires adding token to the place *no more problems*, but the student still needs to meet his tutor again to discuss the result. (Chain *meeting with tutor*–*discussion with tutor*-adding token to place *student* fires again). Only after that student understands the nuances of the problem and the solution well enough.

Then the cycle may repeat again — student ends the process or gets a new problem. If the first case has place, the hammock finishes its execution.

Now we consider the second hammock. This hammock starts with transition *send assignment* firing adding tokens to places *solving assignment* and *submission opened*. Then the hypothetical student has two options: complete his work before deadline (transition *submission*), let the system register his work (transition *deadline*), and then just get his mark by getting through transition *evaluate work*.

If student had not managed to pass his work in time, the system registers this (transition *deadline*) and the submission is closed (disabling transition *submission*). Then he or she has no choice, but to pass the work behind time (transition *commission arranged*), wait for a re-examination (place *preparing for commission*) and go through it (transition *commission*). Independently of the success or the failure of his examination, the student gets his mark — would it be A or F at the transition *evaluate work* firing.

We will not provide the formal definition of a workflow net here, but this model is a sound workflow net, i.e. initial marking has one token in the *start* place. When a token reaches the *end* place there are no other tokens in the net. And the marking with the *end* place marked can always be reached.

IV. Time Petri Nets

In this section we define Time Petri nets (TPN).

We will use the definition of Time Petri net as given in [10]. Time Petri net (TPN) is an extended classic Petri net where each transition t is associated with a time interval $[a_t, b_t]$. If t is enabled, it still cannot fire before a_t time units have elapsed, and it has to fire no later than b_t time units after being enabled, unless it has meanwhile become disabled by the firing of another transition.

Time counting is started from the moment of enabling t and is reset if t has been disabled. a_t is called *"earliest firing time"* of t (short $eft(t)$) and b_t is *"latest firing time"* of t (short $lft(t)$). The firing of a transition does not take any time.

The interval bounds are non-negative rational numbers, and b_t can also be ∞.

Definition 4.1: A Time Petri net (TPN) is a 6-tuple $Z = (P, T, F, V, m_0, I)$ such that

1) the 5-tuple $S(Z) = (P, T, F, V, m_0)$ is a Petri net,
2) $I : T \to Q_0^+ \times (Q_0^+ \cup \infty)$ and for each $t \in T$, with $I(t) = (I1(t), I2(t))$ it holds that $I1(t) \leq I2(t)$

Here $I(t)$ is the time interval of the transition t, during this interval t is ready to fire, I1(t) is eft(t) and I2(t) is lft(t).

Definition 4.2: (p-marking) Let P be the set of all places in a Time Petri net Z. A p-marking in Z is a (total) function $m : P \to \mathbb{N}$.

Definition 4.3: (t-marking) *Let T be the set of all transitions in a time Petri net Z. Any (total) function $h : T \to \mathbb{R}_0^+ \cup \#$ is a t-marking in Z.*

Definition 4.4: Let $Z = (P, T, F, V, m_o, I)$ be a Time Petri net, m a p-marking and h a t-marking in Z. A state in Z is a pair z := (m, h) such that

1) $\forall t((t \in T \wedge t^- > m) \to h(t) = \#)$.

2) $\forall t((t \in T \wedge t^- \leq m) \to h(t) \in \mathbb{R}_0^+ \wedge h(t) \leq lft(t))$.

Claim 1.*Time specification of transitions of an arbitrary time Petri net can be required to be non-negative integers without loss of generality [10]*

V. Reachability graph

Firstly we consider original algorithm proposed in [10]. It is based on the conception of essential states.

Definition 5.1: (modified state change) Let τ be a non-negative real number and $z = (m, h)$ be a state in the Time Petri net Z. It is possible for time τ to elapse in the state z in Z if

$$\forall (t \in T \wedge h(t) \not\to h(t) + \tau \leq lft(t))$$

The elapsing of time τ will change z into the state z' = (m', h') with

1) m':=m,

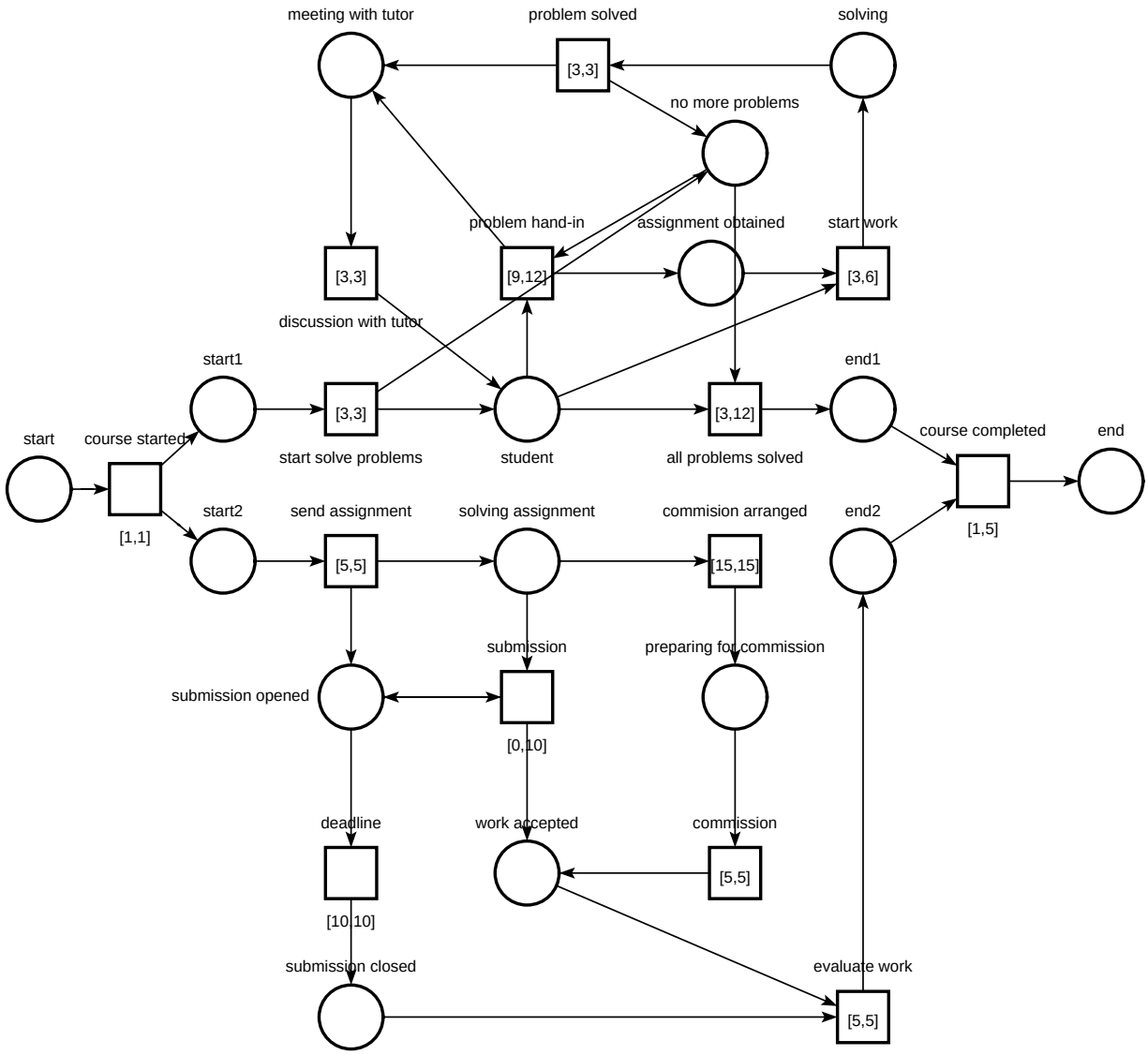

Fig. 1. The workflow net of a course process

2) $\forall t \in T \rightarrow$

$$h'(t) := \begin{cases} \# & \text{if } ift^- > m' \\ eft(t) & t^- \leq m' \wedge lft(t) = \infty \\ & \wedge\ eft(t) < h(t) + \tau \\ h(t) + \tau & \text{otherwise} \end{cases}$$

Definition 5.2: (essential-state) An integer-state z = (m, h) in a Time Petri net Z is called essential-state when Z is defined with the modified firing rule.

Definition 5.3: (reachable essential-state) Let Z be an arbitrary Time Petri net. The set $REIS_Z$ of all reachable essential-states in Z is defined as follows:

$REIS_Z := \{z | z_0 \overset{\sigma(\tau)}{\rightarrow} z\}$, *z is an essential-state and* $\sigma(\tau)$ *is a run feasible in Z*

Set of all reachable essential states of the net carries complete information about this net's behavioural properties. This characteristic makes it possible to construct reachability graph, which includes only essential states (and is much smaller than original reachability graph) and still provides enough information to analyse behavioural properties of the net.

Definition 5.4: (reachability graph for arbitrary Time Petri net) Let $Z = (P, T, F, V, m_0, I)$ be an arbitrary (finite or infinite) Time Petri net with $T = t_1, \ldots, t_n$. The (reduced) reachability graph $RG_Z^{redu} := (W, E, L)$ of Z is the directed graph with edge labels whose set of vertices W, set of edges E and edge labels from $L \subseteq N \times T$ are defined by Algorithm 1.

Algorithm 1: Time Petri net reachability graph

begin
  $R := Z_0$; $W := \emptyset$; $E := \emptyset$;
  while $R \neq \emptyset$) **do**
    select $z = (m, h)$ from R; $R := R - \{z\}$;
    $W := W \cup \{z\}$;
    if $\{t \in T \| t^- \leq m\} \neq \emptyset$ **then**
      if $\{t \in T | t^- \leq m \wedge lft(t) \neq \infty\} \neq \emptyset$ **then**
        Let $k := min\{lft(t) - h(t) | t^- \leq m\}$
        else
          Let
          $k := max\{eft(t) - h(t) | t^- \leq m\}$
    for $time = 0$ ***to*** k **do**
      for $i = 1$ ***to*** n **do**
        if t_i *ready to fire in* $(m, h) + time$ **then**
          Let z' be such that $z \overset{time\, t_i}{\rightarrow} z'$;
          $E := E \cup \{(z, [time, t_i], z')\}$;
          if $z' \notin W$ **then**
            $R := R \cup \{z'\}$

Algorithm 2: Modular construction of Time Petri net reachability graph

begin
  Let H **be the set of hammocks in the net**;
  $X := \emptyset$;
  foreach $h \in H$ **do**
    foreach *Transition* $t \in h$ **do**
      Insert $eft(t)$ into X;
      Insert $lft(t)$ into X;
    GCD_h**=Find GCD(**X**)**;
    foreach *Transition* $t \in h$ **do**
      $T'_h = \{t' | t \in T_h \wedge eft(t') =$
      $eft(t)/GCD_h \wedge lft(t') = lft(t)/GCD_h\}$;
    $h' = (P_h, T'_h, F, V)$
    Apply Algorithm 1 to h

VI. Modular construction of Time Petri net reachability graph

Algorithm constructing reachability graph through essential states demonstrates high performance, when it works with nets, whose time intervals are not very big. But if we multiply all time specifications in the net on the same number, number of essential states increases dramatically. But such scaling does not affect behavioural properties of the net and the set of reachable p-markings of the scaled and the original nets are the same.

There is an evident way to fix the problem — just check if the net can be down-scaled (divided by greatest common divisor) and than analyse this downscaled net. But what if we have the net consisting of two independent components. Both of these components can be down-scaled, if considered independently, but the time specification of these components contains coprime numbers.

Such net can have huge reachability graph, which can not be constructed because of the time or memory restrictions. But the graphs for each of these parts can easily be built via the downscale procedure. And these graphs carry enough information to analyse behavioural properties of the original net, such as boundedness, for example. We propose the way to solve the problem through searching for the detached hammocks, finding greatest common divisor (GCD) for all natural time specification inside them, dividing these specifications on the GCD and than applying reachability graph construction algorithm to each of the found hammocks.

Obtained reachability graphs can be used for performing model checking upon each of them and deduce identified properties for the whole net, but for the moment we consider only receiving reachability graphs.

We applied this technique to our example. The example contains two hammocks. First includes all vertexes and transitions between places *start1* and *end1* inclusively. Second starts at place *start2* and ends with *end2*. We can predict the successful application of the modular approach as the GCD of the first hammock time specifications is 3 and the GCD of the second hammock is 5, which are coprime numbers. Then we applied 2 and received the following results Table I.

The reachability graph of the whole net consists of 208 vertexes and 907 edges. But if we split up the net into 2 hammocks and then construct reachability graph for each of them, then the sum of them will be much lesser than the reachability graph of the whole net. In our case, the original net's reachability graph is not very big, because it is just an example, but even here reduction is significant.

TABLE I. Original and modified algorithms comparison

	Vertexes	Edges
Original (full net)	208	907
Modified (hammock 1)	8	14
Modified (hammock 2)	7	9

VII. Conclusion

In this paper we provided an example Time Petri net that models the process of an academic course. The numeric properties of the transitions time specifications and our modification of the algorithm enabled us decrease time and space requirements for the analysis of the net. We constructed a reachability graph that is much smaller than the original one. The further research directions are to characterize the class of time Petri nets and the behavioural properties for which such technique is applicable.

Acknowledgment

This study was carried out within the National Research University Higher School of Economics' Academic Fund.

References

[1] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux. Comparison of different semantics for time petri nets. In *ATVA*, pages 293–307, 2005.

[2] B. Berthomieu, M. Boyer, and M. Diaz. Time petri nets. In *Petri Nets*, pages 123–161.

[3] H. Genrich and K. Lautenbach. The Analysis of Distributed Systems by means of Predicate/Transition-Nets. In G. Kahn, editor, *Semantics of Concurrent Compilation*, volume 70 of *Lecture Notes in Computer Science*, pages 123–146. Springer-Verlag, Berlin, 1979.

[4] K. Jensen and L. M. Kristensen. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer, 2009.

[5] L. Lamport. What good is temporal logic? In *IFIP Congress*, pages 657–668, 1983.

[6] I. A. Lomazova. Nested petri nets for adaptive process modeling. In *Pillars of computer science*, pages 460–474. Springer, 2008.

[7] I. A. Lomazova. Interacting workflow nets for workflow process re-engineering. *Fundamenta Informaticae*, 101(1):59–70, 2010.

[8] S. S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, 1982.

[9] L. Popova-Zeugmann. *Essential states in time Petri nets*. Citeseer, 1998.

[10] L. Popova-Zeugmann. *Time Petri Nets*. Springer, 2013.

[11] C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. 1974.

[12] M. Schiffers and H. Wedde. Analyzing program solutions of coordination problems by cp-nets. In J. Winkowski, editor, *Mathematical Foundations of Computer Science 1978*, volume 64 of *Lecture Notes in Computer Science*, pages 462–473. Springer Berlin Heidelberg, 1978.