USING TIME PETRI NETS FOR MODELING AND VERIFICATION OF TIMED CONSTRAINED WORKFLOW SYSTEMS

Pedro M. Gonzalez del Foyo, pedro.foyo@poli.usp.br
José Reinaldo Silva, reinaldo@usp.br
Dept. Of Mechatronics, University of São Paulo, Brazil

Abstract. *Time management is a critical component of workflow-based process. Important aspects of time management include: the planning for the workflow process execution on time, estimating workflow execution duration, avoiding deadline violations, and satisfying all time constraints specified. In this paper, we apply Time Petri Nets to model the temporal behavior for workflow systems using TINA as a tool to support the verifications of the activities deadlines.The process is based on the construction of a unique Timed Graph where verification of TCTL formulas is made avoiding the necessity of the construction of a Timed Graph for each formula. The result of this proposal could be used to implement scheduling algorithms in execution time guaranteeing the satisfaction of all external time constraints and activity deadlines. An example is presented to illustrate the method and, finally, a comparison with other approaches for similar problems is included.*

Keywords: *Timed constrained systems, Time Petri Nets, workflow analysis, Model checking*

1. INTRODUCTION

Today, the most critical demand in companies striving to become more competitive is the ability to control the flow of information and work throughout the enterprise in a timely manner. Workflow management systems (WFMSs) improve business processes by automating tasks, getting the right information to the right place for a specific job function, and integrating information in the enterprise (WFMC, 1996). A few years ago, the need to explicit manage time in workflow management systems was identified. Since them, some approaches have been presented to support and manage time constraints associated with processes. This is done mainly in the form of a deadline because many business processes, which are abstracted to workflow, have deadlines.

There are different approaches for modeling temporal aspects in workflow systems. In (Eder et al, 1999) a timed workflow graph is used to model the WFMS and incorporate the temporal constraints. In this work was introduced a technique for scheduling the activities in order to avoid deadline violations. This technique was improved in (Eder et al, 2000) leading to the proposal of a framework called T-WfMC for modeling and building the specification of WFMSs. In (Cicekli-Kesim and Yildirim, 2000) a framework for specifying and executing workflows based on Event Calculus was proposed. This approach uses the first order predicate logic for temporal reasoning. Other formalism has been used to model, specify and analyze workflow systems like control flow graph (Harel, 1987), event-condition action rules (Baral and Lobo, 1997), Petri Nets (van der Aalst, 1998) and enhanced Petri net systems (del Foyo, 2001) but without considering the temporal issues at design or execution times (only the partial order of processes is considered in those cases).

In this paper we propose the use of Time Petri Nets (Merlin and Faber, 1976) for the modeling of workflow management systems in a timely manner. It has been already proved Petri Nets' good expressiveness to represent workflow control semantic. Now we propose the use of Time Petri Nets to represent the remaining aspects: functionality and temporal constrains.

The paper is organized as follows: Section 2 provides basic concepts of Time Petri Nets and TCTL logic for time specifications. Section 3 shows the process of modeling a WFMS and its temporal constrains with an example and introduces a verification method using TINA to obtain the state classes, and from that, a Timed Graph where TCTL formulas will be checked. Section 4 presents the application of these results to the scheduling of workflow. A comparison with others approaches for the same problem is included in Section 4. The method presented here is applied to an example throughout the paper. Following, comments on conclusions and further work is discussed.

2. BACKGROUNDS

In this section we present some background definitions that are useful to understand our proposal. More detailed discussions about these matters could be found in the references.

2.1 Time Petri Nets, terminology and behavior

Merlin's Time Petri nets (TPN) extend Petri nets with temporal intervals associated with transitions, specifying firing delays for transitions.

This section gives a short definition of TPN according to Berthomieu and Diaz, (1991). The notion of state and state classes will be useful to discuss behavioral analysis for TPN, and avoid combinatorial explosion.

A Time Petri Net is a tuple (P, T, B, F, M_o, SIM) where:

- P is a finite non-empty set of places p_i;
- T is a finite non-empty set of transitions t_i;
- B is the backward incidence function
  $B : T \times P \longrightarrow N$;
  where N is the set of nonnegative integers;
- F is the forward incidence function
  $F : T \times P \longrightarrow N$;
- M_o is the initial marking function
  $M_o : P \longrightarrow N$
- SIM is a mapping called static interval
  $SIM : T \longrightarrow Q^* \times (Q^* \cup \infty)$
  where Q^* is the set of positive rational numbers.

The static interval is composed by two positive rational numbers (α, β), where α represents the earlier firing time (EFT) and β the latest firing time (LFT). Assuming that a transition t became enabled at time τ , then t cannot fire before $(\tau + \alpha)$ and no later than $(\tau + \beta)$ unless disabled by firing some other transition.

Time Petri Nets can subsume ordinary Petri Nets or even Timed Petri Nets (Ramchandani, 1974). Using time intervals $(0, \infty)$ in all transitions, Time Petri Nets behave like ordinary Petri Nets, while by using (τ, τ) it reproduces the Timed Petri Nets behavior.

In TPN, the enabling condition of a transition is the same as in Petri Nets. According to the definition of TPN the "enabledness condition" will be:

$$(\forall p)(M(p) \geq B(t_i, p)); \tag{1}$$

Some transitions may be enabled by a marking M, but not all of them may be allowed to fire due to the firing constraints of transitions (EFT's and LFT's). So the "firability condition" depends of two conditions:

1. the transition is enabled, formally expressed by Eq. (1)
2. expresses the fact that enabled transitions may not be fired before its EFT and must be fired before or at its LFT unless another transition fires before, modifying marking M so the transition is no longer enabled.

According to the second condition, a transition t_i enabled by M at absolute time τ could be fired at the firing time θ, iff θ is not smaller than the EFT of transition t_i and not greater than the smallest of the LFT's of all the transitions enabled by marking M, that is: EFT of $t_i \leq \theta \leq$ min{ LFT of t_k} where k ranges over the set of transitions enabled by M. If transition t_i fires, it leads the system to another state, at time $\tau + \theta$.

The state of the system can be defined as a pair $S = (M, I)$ where:

- M is a marking
- I is a firing interval set which is a vector of possible firing times. The number of entries of this vector is given by the number of transitions enabled by marking M. Each entry maps to a pair of time values, the minimum and maximum time that each transition is individually allowed to fire.

The single behavior "transition t_i is firable from state S at time θ and its firing leads to a state S'" will be denoted as:

$$S \overset{(t_i,\theta)}{\longrightarrow} S'$$

A firing schedule will be a sequence of pairs

$$(t_1, \theta_1) \; . \; (t_2, \theta_2) \; .... \; (t_n, \theta_n)$$

in which t_1, t_2, ... ,t_n are transitions and θ_1, θ_2, ...,θ_n are times. This firing schedule is feasible from a state S iff there exist states S_1, S_1, ..., S_n such that:

$$S \overset{(t_1,\theta_1)}{\longrightarrow} S_1 \overset{(t_2,\theta_2)}{\longrightarrow} S_2 - - - - \rightarrow S_{n-1} \overset{(t_n,\theta_n)}{\longrightarrow} S_n$$

The firing rule permits the computing of states and a reachability relation among them. The set of states that are reachable from the initial state, or the set of firing schedules feasible from the initial state, characterizes the behavior of the TPN, the same way that the set of reachable markings characterize the behavior of a Petri net.

In general, using this set of states for analysis purpose is not possible, since this set could be infinite. That is why Berthomieu and Menasche (1983) introduce an enumerative approach for analyzing TPN using what they call state classes.

A state class, denoted by C will be defined by a pair (M, D) where M is a marking and D is a firing domain. The firing domain D will finitely represent the infinite number of firing times possible from marking M, through the set of solutions of a system of inequalities that capture the global timed behavior of the TPN.

The number of state classes will be finite if the underlying Petri net is bounded. Boundeness for TPN has been shown undecidable although a number of sufficient conditions could be stated (Berthomieu and Diaz, 1991). The tool Tina (TIme petri Net Analyzer) proposes several state class space constructions, preserving some properties expressed either in linear time temporal logics (such as LTL) or in branching time temporal logics (such as CTL).

In this paper, we are concerned with a class of properties of great practical interest for which no dedicated construction is proposed by the authors of TINA. That "quantitative" temporal properties may be expressed, for instance, in TCTL logic. However, TINA has other constructions such as atomic state classes (ASCG), which preserve the branching properties like CTL or CTL*. Any graph of classes bisimilar with the state graph of the net preserves all its branching properties, but time information is omitted on this graph in TINA (Berthomieu, 2004). This will be addressed in section 4.

2.2 TCTL logic for time behavior specification and model checking

Timed Computation Tree Logic (TCTL) extends CTL model-checking (Clarke and Emerson, 1986) to the analysis of real time systems whose correctness depend on the magnitudes of timing delays. For specifications, the syntax of CTL was extended to allow quantitative temporal operators. Since the proposal of TCTL logic in Alur et al (1990) it has been used in many applications for model checking approaches in real time systems. TCTL is a branching temporal logic for the specification of dense-time systems.

Let, P be a set of propositions, and Q^* the set of positive rational numbers. The formulas φ of TCTL are inductively defined as follows:

$$\varphi ::= q | I \text{ false } \varphi_1 \rightarrow \varphi_2 | \exists \varphi_1 U_{\sim c} \varphi_2 | \forall \varphi_1 U_{\sim c} \varphi_2$$

where $p \in P$, $c \in Q^*$, and $\sim$ stands for one of the binary relations $<, \leq, =, >, \geq$.

Modal operators like $\Diamond$ and $\Box$ meaning "sometimes" and "always", respectively, can also be used in TCTL formulas.

When TCTL is used as a verification framework (model checking), it is necessary a timed automata (TA) (Hezinger et al, 1992) as a behavior description and a TCTL formula as the specification. Thus, an algorithm should check whether or not the behavior description satisfies the specification. A TA is a finite-state automata equipped with a finite set of clocks that can hold non-negative real-values. In its operation, one transition can be triggered when a corresponding triggering condition is satisfied. Upon being triggered, the automata instantaneously transit from one mode to another and resets some clocks to zero. Between transitions, all clocks increase readings at a uniform rate.

A Timed Automata is a tuple G = $(S, \mu, s_0, E, C, \pi, \tau)$, where

- S is a finite set of nodes;
- $\mu : S \rightarrow P$ is a labeling function assigning to each node the set of true atomic propositions in that node;
- $s_0 \in S$ is the start node;
- $E \subseteq S \times S$ is a set, of edges;
- C is a finite set of clocks;
- $\pi : E \rightarrow 2^C$ tells which clocks should be reseted with each edge.
- τ is a function labeling each edge with an enabling condition built with the use of boolean connectives over the atomic formulas of the form $x \leq c$ or $x \geq c$, where $x \in C$ and $c \in Q^*$

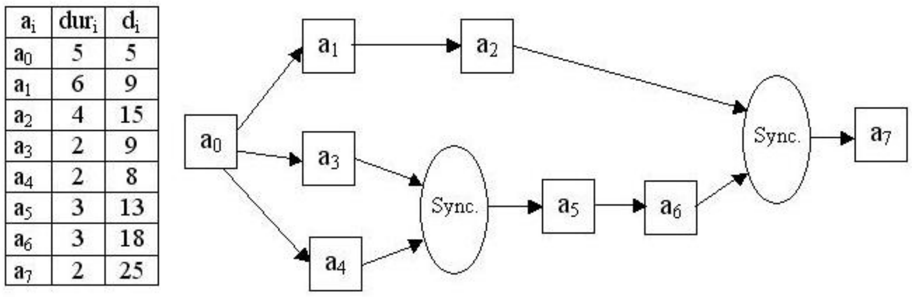
(Lime and Roux, 2006) presents a definition of TCTL for TPN. The only difference with the versions of Alur et al (1990) is that atomic propositions usually associated with states are properties of markings. We use that definition for TCTL model checking in order to construct the formulas that represent the properties to be checked.

3. SPECIFICATION AND VERIFICATION OF WORKFLOW PROCESSES WITH TIME PETRI NETS

Since the introduction of TPN in Merlin and Faber (1976) it has been widely used for the specification and verification of systems in which satisfiability of time constrains is essential, like communication protocols, hardware components or real time systems (Berthomieu, 2003). Even considering TPN a powerful approach for the modeling, specification and verification of real time systems and time constrained systems there are some limitations. TPN models only can be verified using a model checking approach if they are bounded, what means to have a finite state space. In that sense, real time systems have to be modeled by a bounded TPN which often is not possible. Workflow systems have not this problem because they are often (if not always) finite. Therefore, TPN is an useful framework for model, analysis, specification and verification of time constrained workflow systems.

The other limitation is the complexity of behavior and with the size of the modeled systems. Workflow systems are well behaved in the sense that are well structured, that is, composed by control structures as defined in WFMC (1996), (Aalst, 1998), (del Foyo, 2001).

Figure 1 shows an example of workflow. The table on the left side shows a deadline and duration of each activity.



a_i	dur_i	d_i
a_0	5	5
a_1	6	9
a_2	4	15
a_3	2	9
a_4	2	8
a_5	3	13
a_6	3	18
a_7	2	25

Figure 1. Example of workflow: deadlines, durations and temporal constrains.

In this paper we use a TPN to model the same example. Each transition has a time interval attached to them, where the EFT represents the activity duration and the LFT, the maximum time to the completion of the activity that guarantees all system deadlines.

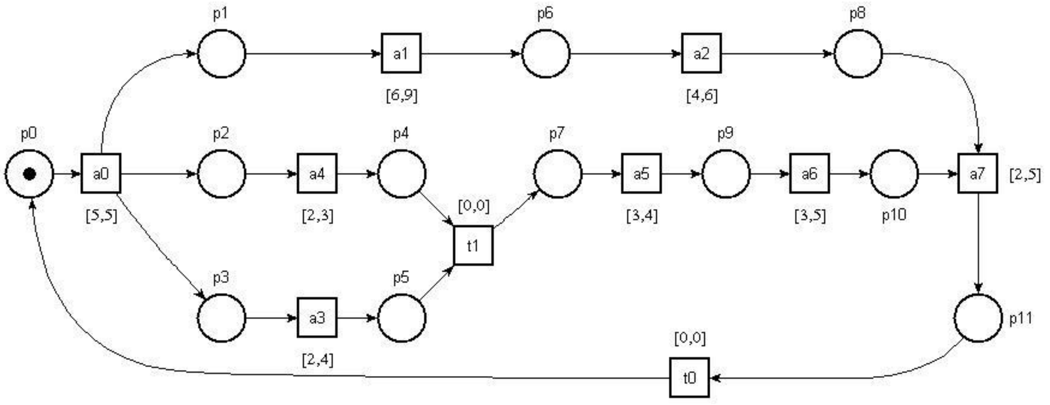


Figure 2. Time Petri net model for the example in Figure 1

In Figure 2, the LFT of each activity was calculated using the critical path approach proposed in Hyun Son and Ho Kim (2000). The method called *Innermost Control Structure First* (ICSF) let us calculate maximum duration of activities regarding its deadlines but cannot deal with time constraints between activities that belong to different branches. The verification process can treat this problem. This issue will be addressed in the next section.

3.1 The verification process

Usually in time constrained workflow systems, time constraints refer to global time or time elapsed between activities. Because of that, only one clock is needed to satisfy the expressiveness of workflow systems. In this paper we are proposing a modification to the Timed Automata in order to simplify the model checking algorithm complexity. It has been proved that for one clock TA the model checking algorithm for TCTL formulas is PSPACE-complete (Laroussinie, 2004).

Then, a modified Timed Automata is a tuple $G = (S, \mu, s_0, A, E, It)$, where

- S is a finite set of nodes;
- $\mu : S \rightarrow P$ is a labeling function assigning to each node the set of true atomic propositions in that node;
- $s_0 \in S$ is the start node;

- A is a set of actions or events.
- $E \subseteq S \times A \times S$ is a set of edges;
- $It : E \longrightarrow Q^* \times (Q^* \cup \infty)$ are the minimum and maximum time elapses to change the current state to the next one through the occurrence of $a \in A$;

Since this modified TA is obtained through the state classes, we will show how to build the TA from the results obtained by TINA.

The state of the system is identified in a TPN as $S = (M, I)$. Let τ be the time when the system achieves the S state. Consider that at least one transition enabled exists at S, then exists a time interval in which S could lead to state S' through the occurrence of that transition. These could be represented as:

$$S \overset{t_1[\theta_1,\theta_2]}{\longrightarrow} S'$$

where $\tau + \theta_1$ is the minimum time in which the system could achieve state S' coming from S through the occurrence of transition t_1 and $\tau + \theta_2$ is the maximum time in which the system could achieve state S' coming from S through the occurrence of transition t_1.

Consider that there is more that one transition enabled at S, so:

$$S \overset{t_2[\theta_3,\theta_4]}{\longrightarrow} S''$$

of course, $\theta_1 \leq \theta_4$ and $\theta_3 \leq \theta_2$. Transitions t_1 and t_2 appear in I, t_1 appears in I'' and t_2 appears in I'.

If we have the reachability graph, the state classes and the static time interval for each transition in the net, we could calculate the time interval It for each edge in the modified TA. State S in TA corresponds to state classes of the TPN, where A is a set of all transitions in a TPN system. The labeling function μ is constructed assigning to each state $s \in S$ the marked places in its equivalent state class.

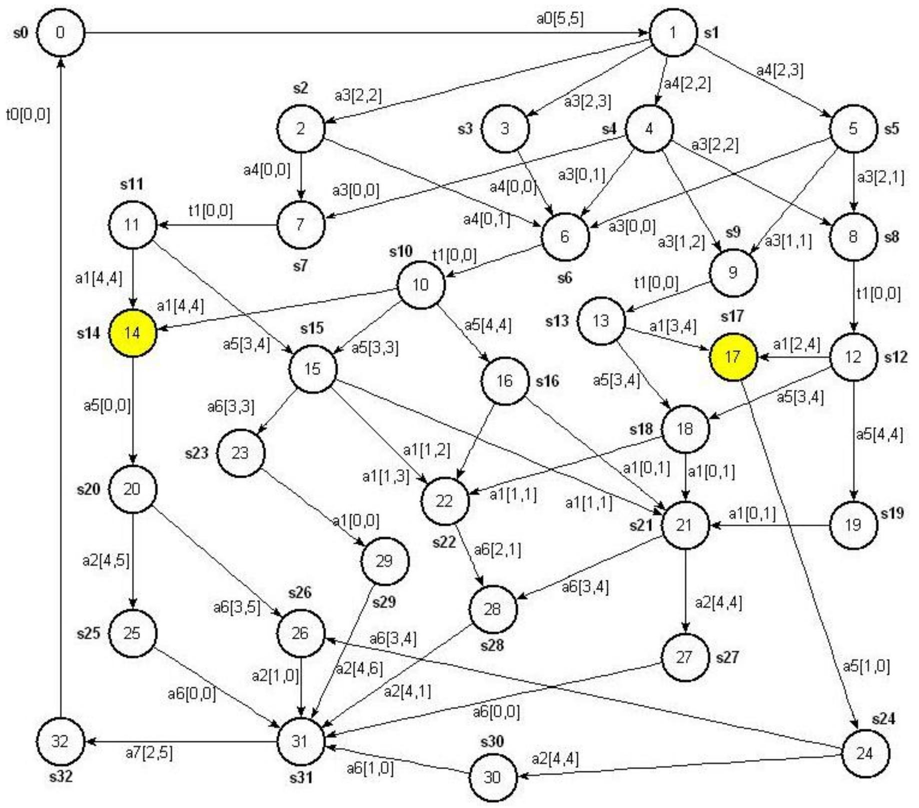


Figure 3. Modified Timed Automata for the TPN in Figure 2

Figure 3 shows the modified TA for the example modeled in Figure 2. To build this TA the results of TINA were used from the options *tools - reachability analysis - state classes preserving (state and CTL* (-U))*. Then, the time interval for each edge of the TA were calculated.

Now we are ready to proceed with verifications. Consider that you want to check if all activities could be completed in 25 time units according to the fixed deadline. The TCTL formula to be checked will be: $\forall\Box_{\leq 25}(p11)$ that means: "p11 is always achieved at least in 25 time units". But it will be desirable to circumvent the global strategy of examining all TA to calculate the $\exists\Diamond_{>25}(\neg p11)$ and obtain $false$. The interpretation of $\exists\Diamond_{>25}(\neg p11)$ will be: " a path exists in which $p11$ is not valid with time greater that 25 time units", of course if the result of this question is *true* the algorithm stops, and the property checked is not valid.

We also could verify this property through and "observer". This approach is known as "TPN + observer" (Toussaint et al, 1997). The problem with that approach is that the "observer" may be as big as the net if the property to be verified involves markings, thus, the number of classes may be squared. Furthermore, each property requires a specific observer, and thus, a new computation of the state class graph (Lime and Roux, 2006). Figure 4 shows the verification of above property with an observer. The result of this computation with TINA shows that $t2$ is a dead transition, therefore the referred property holds.

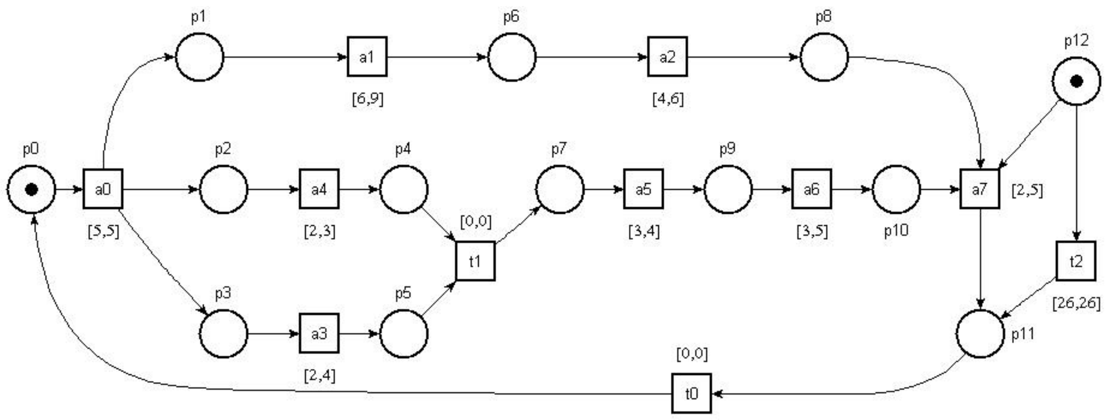


Figure 4. Time Petri Net model with observer to check property $\forall\Box_{\leq 25}(p11)$

Other verifications can be done over the same modified TA. The specification of time constraints could be done through TCTL formulas that contain propositions (places in the TPN). Consider that we have an additional constraint: "$a5$ and $a2$ can't be processed at the same time". There is no precedented order between these two activities, but original deadlines must be preserved.

The formula to be checked is $\exists\Diamond_{>0}(p6 \wedge p7)$ and it fails in all computation paths involving states $s14$ or $s17$. These states are marked yellow in Figure 3.

4. SCHEDULING TIME CONSTRAINED WORKFLOW SYSTEMS

For a TPN model an schedule is a valid firing sequence of transitions that satisfy all timing constraints and leads the system from initial state to one (or many) goal states. Considering $p11$ as a goal state in TPN model in Figure 2, formulas like $\exists\Box_{\leq 25}(p11)$ are enough to guarantee that the system is schedulable. The firing sequence or computational path satisfying this property is in fact an schedule.

Like in (Eder et al, 1999) we define a schedule to be a (more restrictive) TPN in which any combination of activity completion times less than or equal to LFT satisfies all timing constraints. Consequently, as long as activities finish within their ranges, no re-computation is needed for scheduling purposes. Only when an activity finishes outside its range, the schedule for the remaining activities must be recomputed.

Thus, if the property $\forall\Box_{\leq 25}(p11)$ holds, the interval [EFT, LFT] of each activity can be used by an scheduling routine in building time and no runtime re-computation will be required. Only in case of violation of this intervals re-computation will be required in order to re-schedule the remaining activities.

Figure 5 shows a graph with the result of an scheduling algorithm used in (Kafeza and Kamalakar, 2000) for an emergency room example. The time interval associated to each activity indicates minimum and maximum times to start each activity without losing any deadline of the system.

The critical path algorithm used here to obtain the start point for verifications was used in other works such as (Hyun Son and Ho Kim, 2000) and (Eder et al, 1999). Like in (Eder et al, 1999) the valid ranges of execution for each activity are computed at the time of design. The difference is that with our approach is possible add constraints between activities to be executed in parallel as showed in the previous section.

The TPN model of the emergency room example is shown in Figure 6. The model was verified for all constraints. The time interval has a semantic according to TPN, different from that used in T-WfMc graphs.

a_i	d_i	dur_i
a_0	5	5
a_1	15	6
a_2	9	2
a_3	9	2
a_4	9	2
a_5	13	3
a_6	13	3
a_7	15	3
a_8	17	6
a_9	23	1
a_{10}	25	2

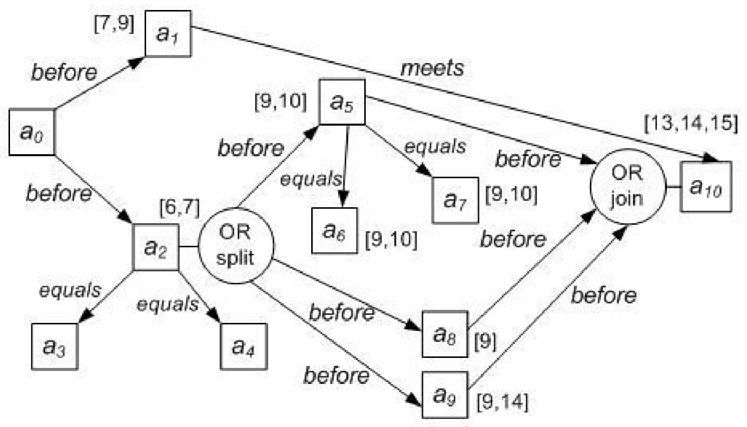


Figure 5. T-WfMc for the emergency room example with time intervals where all timing constraints are satisfied

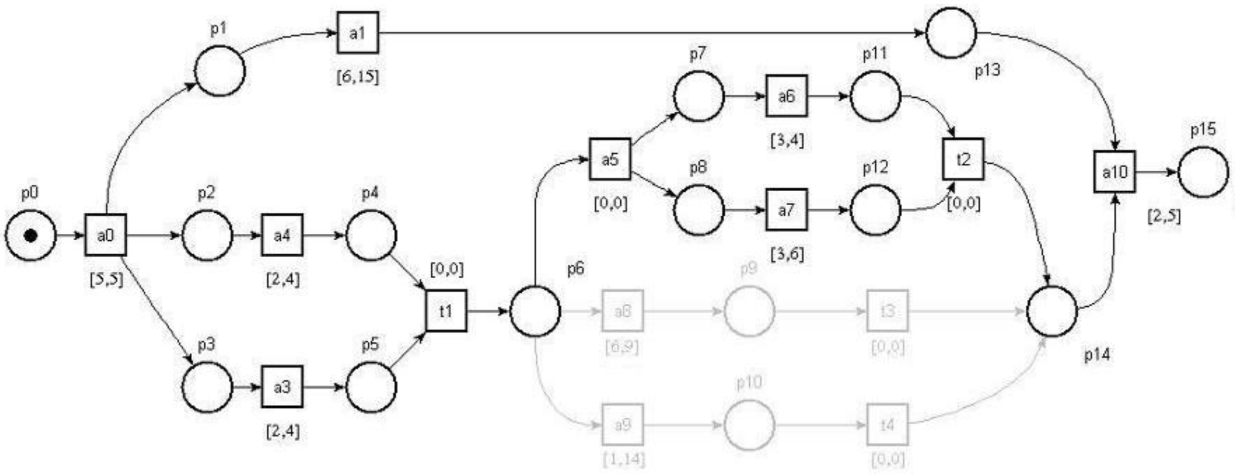


Figure 6. TPN model for the emergency room example with time intervals where all timing constraints are satisfied

5. CONCLUSIONS

In this paper, we have given a method for building the state class graph of a TPN as a TA. This TA uses a single clock and is based on the results of the atomic state classes (ASCG), which preserves the branching properties like CTL or CTL*, offered in TINA. The time interval presented in each edge of the automaton allows to calculate quantitative temporal properties like TCTL; proved to be useful in practical applications. Using this automaton, the state classes and the time intervals need to be computed only once, even if several properties need to be checked.

For scheduling purposes, we show the advantages of verifications of kind $\forall\Box_{\leq c}\varphi$ that guarantee schedulability for all paths if all LFTs of time intervals are achieved. Properties of kind $\exists\Box_{\leq c}\varphi$ also allow to determine an scheduling algorithm but in those cases only the paths evaluated *true* could be included as a fair sequence for the workflow execution. We also show how it is possible to verify properties that involve activities in different branches.

6. ACKNOWLEDGEMENTS

We thank to CAPES to the partial support to the research that result in this article.

7. REFERENCES

Aalst W.M.P. van der, 1998, "The Application of Petri Nets to Workflow Management". *The Journal of Circuits, Systems and Computers*, 8(1):21–66.

Alur R. and Henzinger T.A., 1990, "Real-Time Logics: Complexity and Expressiveness". In *Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 390–401, Washington, D.C., IEEE Computer Society Press.

Alur R., Courcoubetis C., and Dill D. L., 1993, "Model-checking in dense real-time". *Information and Computation*, 104(1):2–34.

Baral C. and Lobo J., 1997, "Formalizing workflows as collections of condition-action rules". In *Dynamics 97, Workshop in ILPS*. Springer-Verlag.

Berthomieu B. and Diaz M., 1991, "Modelling and verification of time dependent systems using time petri nets". *IEEE Transaction on Software Engineering*, 17(3):259–273.

Berthomieu B. and Vernadat F., 2003, "State class constructions for branching analysis of time petri nets". *Lecture Notes in Computer Science*, 2619:442–457.

Berthomieu B., Ribet P. O. and Vernadat F., 2004, "The tool tina - construction of abstract state spaces for petri nets and time petri nets". *Int. J. Prod. res.*, 42(14):2741–2756.

Cicekli-Kesim N. and Yildirim Y., 2000, "Formalizing workflows using the event calculus". In *DEXA 2000*, volume 1873. Springer Verlag.

Clarke E. M., Emerson E. A. and Sistla A. P., 1986, "Automatic verification of finite state concurrent systems using temporal logic specifications". *ACM transactions on Programming Languages and Systems*, 8(2):244–263.

González del Foyo P. M. , 2001, "Ghenesys: Uma rede estendida orientada a objetos para projeto de sistemas discretos".(in Portugueze) Master's thesis, Escola Politecnica da Universidade de São Paulo, Brasil.

González del Foyo P. M. and Silva J. R., 2003, "Towars a unified view of petri nets and object oriented modeling". In *Proceedings of the 17th Brazilian Congress of Mechanical Engineering*, volume 1, Sao Paulo, Brazil, ABCM.

Eder J., Grubber W., and Panagos E., 2000, "Temporal modeling of workflows with conditional execution paths". In *DEXA 2000*, volume 1873. Springer Verlag.

Eder J., Panagos E. and Rabinovich M., 1999, "Time constraints in workflow systems". In *CaiSE 99*, volume 1626, pages 323+. Springer Verlag.

Emerson, E. A., Jutla C. S. and Sistla, A. P., 1993, "On model-checking for fragments of μ-calculus"., *CAV*, pp. 385–396.

Haimowitz I., Farley J., Fields G.S., Stillman J. and Vivier B., 1996, "Temporal reasoning for automated workflow in health care enterprises". In *Electronic Commerce: Current Research Issues and Applications*. Springer-Verlag.

Harel D., 1987, "Statecharts: A visual formalism for complex systems". *Science of Computer Programming*, 8(3):231–274.

Henzinger T. A., Nicollin X., Sifakis J. and Yovine S., 1992, "Symbolic Model Checking for Real-Time Systems". In *7th. Symposium of Logics in Computer Science*, pages 394–406, Santa-Cruz, California, IEEE Computer Scienty Press.

Hyun Son Jin and Ho Kim Myoung, 2000, "Finding the critical path in a time-constrained workflow". *rtcsa*, 00:102.

Kafeza E. and Kamalakar K., 2000, "Gaining control over time in workflow management applications". In *DEXA 2000*, volume 1873. Springer Verlag.

Laroussinie F., Markey N. and Schnoebelen Ph., 2004, "Model checking timed automata with one or two clocks", Tech Report.

Lime, D. and Roux, O. H., 2006, "Model checking of time petri nets using the state class timed automaton", *Discrete Event Dyn Syst* **16**: 179–206.

Ramchandani C., 1974, "Analysis of asynchronous concurrent systems by timed petri nets". Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

Toussaint, J., Simonot-Lion, F.and Thomesse, J.-P., 1997, "Time constraints verification methods based on time petri nets", *FTDCS '97: Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97)*, IEEE Computer Society, Washington, DC, USA, p. 262.

WFMC, 1996, "Workflow management coalition terminology and glossary".

8. Responsibility notice