Modelling and Analysing Real Time System Specifications using Time Stream Petri Nets

Franco Cicirelli, Angelo Furfaro, Libero Nigro *

* Laboratorio di Ingegneria del Software,
Dipartimento di Elettronica Informatica e Sistemistica,
Università della Calabria, 87036 Rende (CS) - Italy
(e-mail: f.cicirelli,a.furfaro@deis.unical.it, l.nigro@unical.it)

Abstract: This paper describes an approach to modelling and analysis of real-time systems which is based on the Time Stream Petri Nets (TSPNs) formalism. The work argues that although TSPNs were originally proposed for modelling multimedia/hypermedia systems, they are well suited for expressing timing constraints in general time-dependent systems. The approach is assisted by some developed tools, based on discrete-event simulation and on model checking in terms of Uppaal timed automata, which permit temporal analysis and in particular schedulability analysis of real-time system specifications. The paper introduces the TSPN formalism and describes an implemented transformation in Uppaal. The modelling/analysis approach is demonstrated by an example. Finally, an indication of further work is given in the conclusions.

Keywords: Modelling, timing constraints, real-time systems, schedulability analysis, Time Stream Petri Nets, simulation, model checking, timed automata.

1. INTRODUCTION

Time Petri nets (TPNs) (Merlin and Farber, 1976) have been proven to be a very convenient formal tool for expressing timing constraints in time-dependent systems such as real-time systems and communication protocols. TPNs associate transitions with time pairs instead of single delays as in timed Petri nets (Ramchandani, 1974).

Property analysis of time-dependent systems abstracted as TPNs has mainly been based on an adaptation of well-known reachability techniques of Petri nets (Murata, 1989) as shown in (Berthomieu and Diaz, 1991; Berthomieu et al., 2004; Bucci and Vicario, 1995; Vicario, 2001). In these approaches a time reachability graph (or tree) is built which provides a representation of the complete dynamic behaviour of a TPN, based on interleaved semantics. Nodes are state classes holding a net marking and a firing domain (inequalities system) reflecting timing constraints, and edges are labelled with firing transitions. Similar approaches were also achieved by mapping TPNs onto timed automata (Ahu and Dill, 1994; Behmann et al., 2004) with model checking, as witnessed in (Cassez and Roux, 2006; Cicirelli et al., 2007a).

As pointed out in (Tsai et al., 1995; Xu et al., 2002) real-time system specifications require in general not only reachability analysis but also schedulability analysis, which is more concerned with assessing timing constraints of task executions and then of transition sequences rather than markings or states. A specific analytical method for schedulability analysis of TPNs is proposed in (Xu et al., 2002). A schedulability analysis approach where TPNs are modelled as Uppaal timed automata was experimented in (Furfaro and Nigro, 2007).

This paper reports current authors’ work on schedulability analysis of real-time system specifications. An original approach and supporting tools are developed which are based specifically on Time Stream Petri Nets (TSPNs) (Sénac et al., 1996; Boyer and Díaz, 1999). Although TSPNs were originally proposed for modelling and property analysis of multimedia/hypermedia systems, this work argues that TSPNs can also be used for timing constraints specification and analysis of general time-dependent systems. TSPN are more powerful than TPNs (Boyer and Díaz, 1999). A basic enhancement with respect to TPNs concerns the possibility of explicitly expressing timing constraints of tasks and then detecting constraint violations both at the task level and at the activity (i.e., single transition firing) level. Timing constraints have the form of time pairs and are associated with arcs only. A weak synchronization model applies to arcs but, as in TPNs, a strong synchronization model is associated with transition firings. Transitions can be annotated with one of several firing rules which give great flexibility to the modeler. TSPNs are capable of supporting the formulation of timing constraints of the kind permitted, for example, by Timing Constraint Petri Nets (TCPNs) (Bai et al., 1995), which are a more complex modelling tool than TPNs. TCPNs differ from TPNs for associating time pairs and durational information to places and transitions and for adopting a weak firing model for transitions.

The rest of this paper is structured as follows. First definitions and semantics of TSPNs are introduced. Then a case study concerned with a patient monitoring system is presented, with the goal of illustrating the expression of timing constraints in real-time specifications. After that, an approach which transforms a TSPN model into a Uppaal model for model checking is discussed. This is
followed by thorough verification of the case study. Finally, conclusions are drawn with an indication of on-going and future work.

2. TIME STREAM PETRI NETS

The following recapitulates TSPNs definitions and their semantics.

A TSPN is a tuple $\langle P, T, B, F, I_{nh}, M_0, IM, SYN, MA \rangle$ where:
- $P$ is a finite nonempty set of places and $T$ is a finite nonempty set of transitions;
- $B$ is the backward incidence function, $B: P \times T \to \mathbb{N}$ and $F$ is the forward incidence function, $F: P \times T \to \mathbb{N}$, where $\mathbb{N}$ denotes the set of natural integers;
- $I_{nh}$ is the set of inhibitor arcs, $I_{nh} \subset P \times T$ where $(P, T) \in I_{nh} \Rightarrow B(p, t) = 0$;
- $M_0$ is the initial marking function, $M_0: P \to \mathbb{N}$, which associates with each place a number of tokens;
- $IM$ is a function which associates with each arc, incoming to a transition, an interval defining its static temporal validity interval. $IM: A \to Q^+ \times (Q^+ \cup \{\infty\})$, where $Q^+$ represents the set of nonnegative rational values, $A = \{a = (p, t) \in P \times T | B(p, t) \neq 0 \lor (p, t) \in I_{nh}\}$ is the set of all incoming arcs and $IM(a) = [t_{min}(a), t_{max}(a)]$ is such that $t_{min}(a) \leq t_{max}(a)$;
- $SYN$ is a typing function, which associates each transition with a firing rule: $SYN: T \to \{And, Weak - And, Or, Strong - Or, Master, Or - Master, And - Master, Strong - Master, Weak - Master, Pure - And\}$
- $MA$ is a function that associates a master arc to each transition whose firing rule requires it. $MA: T_m \to A$, where $T_m = \{t \in T | SYN(t) \in \{Master, Or - Master, And - Master, Strong - Master, Weak - Master\}\}$

2.1 TSPN semantics

The marking of a TSPN is a function $M: P \to \mathbb{N}$ that associates each place with a number of tokens. An arc $a = (p, t) \in A$ is enabled by a marking $M$ iff $a \in I_{nh} \Rightarrow M(p) = 0 \land a \notin I_{nh} \Rightarrow M(p) \geq B(p, t)$. The set of arcs enabled by a marking $M$ is denoted by $enArc(M)$. The set of incoming arcs of a transition $t$ is denoted as $A(t)$. A transition $t$ is enabled by the marking $M$ iff $A(t) \subseteq enArc(M)$. The set of transitions which are enabled by a marking $M$ is denoted as $enabled(M)$. The set of input places of a transition constitutes its pre-set. The set of output places of a transition is its post-set. The state of a TSPN is defined as a pair $(M, I)$ where $M$ is a marking and $I$ is a mapping which associates each arc enabled by $M$ with a dynamic temporal validity interval, i.e. $I: enArc(M) \to Q^+ \times (Q^+ \cup \{\infty\})$, $I(a) = [x(a), y(a)]$. Both the absolute or relative time model can be used with TSPNs. In the following, the relative time model is assumed. The initial state of the system is given by $(M_0, I_0)$, where $\forall a \in enArc(M_0)$ $I_0(a) = IM(a)$. A transition $t_f$ is said to be fireable at relative time $\theta$ from state $(M, I)$ if $t_f$ is enabled by $M$ and $\text{low}(t_f) \leq \theta \leq \text{up}(t_f)$, where $\text{low}(t)$ and $\text{up}(t)$ are determined as follows:

$$\text{low}(t) = \min_{t \in \text{enabled}(M)} \text{up}(t)$$

Let $t$ be a transition fireable from state $S = (M, I)$. The state $S' = (M', I')$ reached by firing $t$ at relative time $\theta$, is computed as follows:

$$S' = \{ (M, I) \}$$

where $a_m = MA(t)$ if $t \in T_m$.

A transition $t'$ enabled in $M$ can lose its enabling during the atomic firing process of $t$ either in the intermediate marking $M$ or in the reached marking $M'$. It is said non-persistent to the firing of $t$. On the contrary, a persistent transition (which is enabled in $M$) keeps its enabling during the whole firing process of $t$. A transition is said newly enabled if it was not enabled in $M$ or in $M'$ but it is enabled in $M'$. While Time Petri Nets (Merlin and Farber, 1976) have only a strong semantics model for transition firings, TSPNs adopt a more flexible interpretation because timing constraints of arcs can be violated if the firing rule of the related transition allows it. In other words, a valid firing interval for an enabled transition may exist also in the case the timing constraints of some of the incoming arcs are violated. Once a valid timing interval is found for a transition, it constitutes a strong constraint on its firing. Of course, arc violations can determine transition violation if no valid timing interval is possible for the transition. As a consequence, TPSNs use a weak synchronization model for arcs but a strong one for transitions. As it is usual in Time Petri Nets (Berthomieu and Diaz, 1991; Vicario, 2001) a multiple enabled transition is assumed to fire its enabling one at a time (single server semantics). After its own firing, a transition $t$ which is still enabled is considered newly enabled.
2.2 Notes on the synchronization rules

The TSPN synchronization rules (see also Fig. 1) can be driven by (a) the latest arriving process (And, Pure-And, Weak-And, And-Master) where the last arc that reaches the lower bound of its temporal interval allows the firing of its related transition, (b) the earliest arriving process (Or, Strong-Or, Or-Master) where the first arc which reaches its lower bound permits the firing of its related transition, and (c) the arriving of a statically selected process (Master, Strong-Master, Weak-Master), i.e. the transition can only fire when its master arc reaches the lower bound of its associated temporal interval.

Differences between the And and Pure-And firing rules are useful to point out. When the temporal intervals overlap, they behave the same way: the transition can only fire in the intersection of the intervals. In the case some intervals are disjoint, at least one arc is violated when a process gets its minimum bound. In this situation, the And firing rule permits one single synchronization point which coincides with the lower bound of the latest arriving process. In other words, under the And firing rule a transition can always fire, but with the Pure-And the firing is impossible to occur when the intersection is void.

In the master scenario, Master, Or-Master and Strong-Master are violated when the master arc is violated. In the case of a Strong-Master transition with a non master arc already violated, there exists one single synchronization point which coincides with the lower bound of the dynamic interval of the master arc. Instead, a Weak-Master and And-Master can never be violated.

In any case, a transition can fire when it is both logically enabled (in the sense of classical Petri nets) and temporally ready (i.e. the lower bound of its dynamic temporal interval defined according to the adopted synchronization rule, has been reached). For simplicity, Fig. 1 mirrors the absolute time model: the temporal validity intervals of the three arcs are absolutized with respect to the absolute time instants \( t_1 \), \( t_2 \) and \( t_3 \) at which the arcs get logically enabled.

3. A REAL-TIME SYSTEM SPECIFICATION

This section illustrates through an example the capabilities offered by TSPNs for the expression of timing constraints in real-time system specifications. The modeling example, adapted from (Tsai et al., 1995), is concerned with a patient monitoring system (PMS). The purpose of PMS consists in monitoring the pulse rate and the level of oxygen saturation, detected by means of suitable sensors applied on the skin of a patient, for real-time detection of abnormal situations that may be dangerous for the patient’s health.

PMS is made of three distinct components: a sampling subsystem, a signal analyzer and an alarm. The sampling subsystem periodically samples the patient’s pulse, processes the achieved signal and then sends it to the analyzer. The sampling component is equipped by three processors which work at different speed and are able to elaborate the sampled data with different levels of accuracy. Two of these processors are also employed for carrying out other computation activities, while the third one is dedicated to sample processing only. Once a signal has been built from the sampled data it is asynchronously sent to the analyzer. The analyzer component is in charge of analyzing the signal by searching for abnormal shapes and of plotting it on the onboard display. In the case an abnormal situation happens, it synchronously sends an activating message to the alarm component. The alarm remains active until someone turns it off. The PMS model is split into three submodels, respectively associated with sampling, analyzer and alarm components, which are linked to one another by reference/actual interface places. A reference place is shown as a dashed circle and has the same name of its actual place in the partner sub model. Fig. 2 shows a TSPN of the sampling subsystem. Where not explicitly stated, the Pure-And synchronization rule for transitions and a time constraint \([0, \infty]\) for arcs incoming to transitions, are assumed by default. Master arcs are depicted with a bold line.

Transition \( RTSA \) fires each 30 time units modeling the start of a sampling phase. A sampling phase lasts 20 time units and its completion is mirrored by the firing of transition \( SAM \) which produces a token into place \( ISR \). The ISR marking models the presence of sampled data waiting to be processed. Places \( P1R \), \( P2R \) and \( P3R \) reflect the readiness respectively of one of the three processors. The first processor produces the most accurate signal from the sampled data. However, to achieve this result the executed routine needs more processing time with respect to those used by the other two processors. The second processor produces signals with intermediate accuracy, while the third adopts a faster but less precise routine. The computation time of the three routines are respectively modeled by the timing constraints of the arcs incoming to transitions \( T1EP \), \( T2EP \) and \( T3EP \). The following policy for selecting the processor is adopted: the first processor is used only if it is available when the sampled data becomes ready, the second processor is preferred in the case it is found ready after 1 time unit from when the data are produced, the processor three
is otherwise used. This policy is mirrored in Fig. 2 by the Master synchronization rule for transition T1SP and T2SP and the And rule for T3SP. The Master rule with a punctual interval on the master arc ensures that if the transition is not enabled at the prescribed time the arc becomes violated and the transition is no more executable until the token in ISR is consumed. The presence of a token in SGR models the fact that a signal has been generated from the sampled data. After being created, a signal will be ready after 2 time units and must be transmitted within 5 time units. Signal transmission takes 6 time units. A timing constraint exists in the overall task of the sampling subsystem which requires that from the time the sampled data is ready to be processed no more than 15 time units can elapse before the signal is being transmitted to the analyzer. This constraint is modeled by the timing specification of the arc incoming to RFS from place TC1.

Fig. 3 shows a TSPN sub model for the analyzer component. Transition RSS mirrors the reception of a signal which takes 5 time units. After the signal is received, the analyzer employs 20 time units for calculating the figure to be displayed (transition CAL). Then the analyzer starts two activities in parallel. The first one consists in refreshing the display background and it takes 10 time units. The second activity concerns the detection of an abnormal situation. In the normal case, the analyzer takes from 8 to 10 time units for completing the detection activity (transition PNS). The use of the Master rule for PNS allows to express, through the time pair of the master arc, a duration constraint. The analyzer takes 5 time units to assess the occurrence of an abnormal situation. In this case, it sends a message to the alarm component (transition SND) and then waits for the arrival of an acknowledgment. Sending a message to the alarm components takes 5 time units and it must be processed by the alarm before 20 time units are elapsed (transition REC in Fig. 4). The use of the Strong-Master rule for transition REC ensures that its firing may only occur between 5 and 20 time units from when a token appears into MGR and as soon as a token arrives into ALR, meaning that the alarm component is ready. Similar timing constraints exist for the acknowledgement reception (transition RAK in Fig. 3). Both in the normal and abnormal situations, drawing the signal plot takes 10 time units (transition DNS and DAS). A task deadline of 50 time units exists from the time a signal is received by the analyzer until the signal is up to be displayed (see timing constraints on both arcs outgoing from TC2) under normal or abnormal situations.

The sub model of the alarm component is depicted in Fig. 4. Transition SAK models the sending of an acknowledgment to the analyzer. It takes 5 time units to initialize the first alarm notification (transition FAN) and after that the alarm component is ready again. The alarm notification is played repeatedly until someone turns it off (transition TOA). From when an abnormal signal is detected it can elapse at most 25 time units before the first alarm notification is made (see timing constraint on the arc outgoing from TC3).
Fig. 5. Common Arc template

Fig. 6. Basic transition template

4. TEMPORAL VERIFICATION USING UPPAAL

Both a discrete-event simulator (using the absolute time model) and a translation approach based on some reusable process templates and data structures in the context of the Uppaal toolbox (Behrmann et al., 2004) (thus using the relative time model) for property verification through model checking were prototyped. In the following, the design issues of transforming a TSPN model into Uppaal timed automata are discussed. The 4.0.8 version of Uppaal was used in the experiments.

Basic templates are associated with transitions and input arcs of a TSPN model. Only arc templates have clocks. Transitions, though, share clocks with their input arcs and use them according to the firing rules. A few global data structures hold the topology information of a TSPN model, and are the backward (B) and forward (F) matrices of the underlying Petri net. Dynamic status information of a model are stored in the marking vector (M), the matrix of arc states (States) and input arcs vector clock (x). As an arc dynamically changes its state (see Fig. 5), it updates the States data structure accordingly through the status() function. The implementation owes to the possibility of capturing useful computations into C-like functions which can be local to specific templates or global to the system.

Common functions include checking the enabling status separately for transitions and arcs, doing the withdraw and deposit phases of a transition firing, and so forth. A fundamental template is the Arc template (Fig. 5). An arc receives as parameters its relevant transition id, its clock and bounds of its temporal validity interval.

A number of transition templates was developed which are distinguished by the associated synchronization rule and number of input arcs. The most simple transition template is the Basic template (see Fig. 6) which has one single input arc. Its parameters include the unique transition id, and the clock and bounds of the input arc. A basic transition can never be violated.

From Fig. 5 and Fig. 6 it can be understood the timing model of the proposed approach. A transition starts in the not enabled location (N). When both enabled and ready, the transition passes to the firing location (F). The actual firing is constrained to occur within or at the upper bound of the transition dynamic firing interval. The upper bound is mirrored by the invariant attached to F. When the upper bound is infinite, the transition can remain in F, provided it is not disabled meanwhile, an arbitrary amount of time.

The firing process is atomic and consists of two phases: withdrawal of tokens from the input places and deposit of tokens in the output places. As in (Furfaro and Nigro, 2007) the firing process is realized with the help of a broadcast channel here named check. This way, yet at the end of the first withdrawal phase (location W) is possible to influence conflicting transitions which can exit the F location and return back to the N location. The second phase (deposit of tokens, location D) can possibly enable new transitions, or still disable under firing transitions due to the use of inhibitor arcs. The check channel is also eared by arcs which, due to the token game, can become enabled/disabled. A model is bootstrapped by a Starter template which sends an initial check thus causing enabled arcs to enter the E location. After this first communication, the Starter becomes idle and takes no further part in the behaviour of the model.

In a translated TSPN model, time can elapse in the F location of under firing transitions or in the E (enabled) location of arcs waiting for the lower bound of their temporal interval to be reached. In the case an arc gains the ready location R, it sends a broadcast communication through the channel check asking both transitions and other arcs to check their status. When receiving a check, an enabled and ready transition can move from N to F. From the locations E and R, an arc immediately re-enters its disabled status (location D) as soon as it becomes disabled or the just fired transition is its associated transition. When moving to the E location, the arc clock is obviously reset. As a consequence of the above timing model and also considering the atomic firing process of transitions, it is not possible for a TSPN transition to re-gain the F status during current firing (see Fig. 6).

A subtle point concerns the ready status of an arc. In reality the R location is the fusion of two states: readiness and violation. Violation is uniquely identified by the arc being in R and its clock which exceeds the upper bound of the arc dynamic temporal interval.

The above behaviour is also observed by the other types of transition templates. Names of these templates are AndX (X>1), PureAndX (X>1), Master, StrongMasterX (X>1) etc. Of course, an And1, PureAnd1 etc. is already reproduced by the Basic template. One single Master template serves all cases where a transition, having one or multiple input arcs, follows the master synchronization rule. The Master template, shown in Fig. 7, receives as parameters the clock and bounds of the master arc, which drive the temporal behaviour of the transition. The only difference by Master and Basic templates concerns the fact that a master transition can be violated (location V in Fig. 7). As a master transition finds itself enabled and ready, it moves to a committed location which is a decision point. From that location either the F location...
Fig. 7. Master transition template

Fig. 8. The And2 transition template

Fig. 9. The PureAnd2 transition template

is reached or the V location is entered, depending on the arc clock value.

Fig. 8 and Fig. 9 illustrate respectively the And2 and PureAnd2 templates. They can be easily adjusted to cope with more than 2 input arcs. It is recalled that an And transition can never be violated, but a PureAnd obviously can. An And2 transition gets the ids of the two input arcs together with the clock and bounds of the relevant temporal intervals. The same occurs for a PureAnd2 etc. In the And2 template it should be noted that in the case there exists some violated arc, and the transition is enabled and ready, it immediately fires (single point of synchronization).

Fig. 10 shows the StrongMaster2 template. The F location is entered in the case the two dynamic intervals of master arc and second arc overlap (location F), i.e. they both are not violated. In the case the master arc is violated, the transition is violated and reaches the V location. When the non master arc is violated, according to the Strong-Master synchronization rule, one single point of synchronization exists which corresponds to the lower bound of the dynamic temporal interval of the master arc. Therefore, in this case, transition firing is immediate.

Fig. 10. The StrongMaster2 transition template

Fig. 11. A critical timing situation

Templates corresponding to the other firing rules were also built, but here are not reproduced for brevity. The Or related templates (Or, Weak-And and Weak-Master) are tricky due to Uppaal limitation of not allowing disjunctions of clock comparisons as invariants or guards (because they would create non convex zones). As a consequence, each transition template is accompanied by a few helper templates which synchronize with the transition so as to extend the firing period to the maximum upper bound of arc temporal intervals.

As a final remark, it is worth considering in which way the above described implementation strategy solves the “last time point problem” implied by TSPN semantics. The problem is summarized in the TSPN model in Fig. 11 (similar constraints exist in the PMS model shown in the previous section). Transition T1 adopts the Master rule and the master arc is P2-T1. Transition T0 can use the Pure-And rule (but the Basic template suffices). At time 5 transition T0 should fire and at the same time the arc P2-T1 has to decide if it is violated or not. Due to nondeterminism the arc could conclude it is violated because transition T1 is still disabled, but if the event of T0 firing precedes that of violation check of the arc P2-T1, then at time 5 transition T1 gets enabled and the arc detects it is not actually violated.

The problem is automatically solved by the design of the Arc template (see again Fig. 5) which guarantees an arc remains in the uncertainty location R, if not disabled, even when time is strictly beyond the upper bound of its temporal interval. This behaviour in turn ensures that effectively all actions which occur at a given time are completed before the meaning of R possibly changes to violated.

5. ANALYSIS OF THE PMS MODEL

The Uppaal realizations were used to check the temporal correctness of the patient monitoring system example presented earlier in this paper. Since the sampler is asynchronously linked to the analyzer component and the latter is synchronously connected to the alarm subsystem, the analysis activities were carried out modularly by separat-
ing the sampler verification from that of the combination of analyzer and alarm subsystems. An interest in explicitly modelling task timing constraints relates to the fact that in many cases the analysis can reduce to checking for the existence of some arc or transition violations. Besides checking for the absence of deadlocks, the sampler sub-model was verified by issuing the following queries. First of all the marking vector was declared as int[0,1] M[P], P being the number of places, to reflect the assumption that the model is 1-bounded. Any violation of this constraint would stop the verifier for an illegal assignment to a variable. No stop was observed during the analysis. A first query issued was:

A<> ss g.W

which checks that transition SSG in Fig. 2 which sends a signal corresponding to last sampled and processed data to the analyzer is eventually fired. The query was found satisfied. A more explicit query was based on the leads-to operator:

\[ \text{rtsa.W} \rightarrow ss g.W \]

that checks that after each firing of the RTSA transition which begins a period, it is eventually followed by a firing of the SSG transition. The query was found satisfied.

The task deadline of 15 time units, i.e. that after each firing of transition SAM which starts current task of the sampler, transition RFS which terminates the task is eventually executed within 15 time units, was verified by the query:

\[ E< (aTC1\_RFS.R \&\& x[aTC1\_RFS]>15) \]

which checks if there exists a state in the state graph in which the arc aTC1\_RFS is found violated (meaning a deadline missing), was found not satisfied. The same safety property was checked by the query:

\[ E< rfs.V \]

which asks the verifier if exists any state in which the RFS transition is violated. The query was not satisfied. The combination analyzer and alarm subsystem was similarly verified. The same provision discussed above for the sampler was adopted for checking the 1-bounded behaviour of places of analyzer and alarm. Recalling that at each reception of a signal from the sampler, it must follow either a normal behaviour (firing of transition DNS in Fig. 3) or an abnormal one (firing of DAS transition in Fig. 3), it was issued the query:

\[ r s s .W \rightarrow (d n s .W \&\& d a s .W) \]

which was found satisfied. The abnormally behaviour which involves synchronous communication between analyzer and the alarm subsystems, was first functionally checked by:

\[ a s d .W \rightarrow d a s .W \]

which was found satisfied.

Violations of task timing constraints attached to the arcs TC2\_DNS and TC2\_DAS in the analyzer, and TC3\_FAN in the alarm component, were checked by the query:

\[ E< (aTC3\_FAN.R \&\& x[aTC3\_FAN]>25) \]
\[ (aTC2\_DNS.R \&\& x[aTC2\_DNS]>50) \]
\[ (aTC2\_DAS.R \&\& x[aTC2\_DAS]>50) \]

which was found not satisfied.

In the light of the above results, the PMS system was found temporally correct. The same results were also confirmed by the discrete-event simulator. The verification experiments were carried out on a WinXP Pentium IV, 3.2GHz, 3GB RAM. To give an idea of the execution performance, one single query on the sampler model lasts in about 10 sec.

6. CONCLUSIONS AND OUT-LOOK

With respect to Time Petri Nets (TPNs) (Merlin and Farber, 1976), Time Stream Petri Nets (TSPNs) (Boyer and Diaz, 1999) offer the advantage of directly expressing in a model timing constraints both at the task level and at the single activity level within tasks. This paper introduces an approach to TSPN modeling and analysis of real-time system specifications and describes a transformation of a TSPN model into the terms of Uppaal timed automata, for exhaustive verification of temporal properties. Property assessment of large models can be based on a discrete-event simulator. The paper illustrates the practical application of the approach through an example.

On-going work is directed at:

- expanding the library of reusable transition templates
- optimizing the transformation process into Uppaal so as to (possibly) improve analysis performance
- extending the TPN/Designer toolbox (Carullo et al., 2003) so as to allow editing, simulation and automatic generation of the Uppaal code from a graphical TSPN model
- porting the discrete-event simulator of TSPN to the distributed context (Cicirelli et al., 2007b) in order to support large model analysis.

REFERENCES


