Model-based Prediction of Complex Multimedia/Hypermedia Systems

Franco Cicirelli, Libero Nigro, Francesco Pupo
Laboratorio di Ingegneria del Software
Dipartimento di Elettronica Informatica e Sistemistica
Università della Calabria - 87036 Rende (CS) – Italy
Email: f.cicirelli@deis.unical.it, {l.nigro,f.pupo}@unical.it

***Abstract*–This work develops a model-based approach to the specification and analysis of the functional/temporal behavior of complex multimedia/ hypermedia systems. The approach is centered on Time Stream Petri Nets (TSPN) as the authoring formal language and on UPPAAL as the target tool for model checking activities. A structural translation from TSPN to UPPAAL timed automata (TA) was recently defined in the form of a reusable TA library. The library was proven to be timed bisimilar to TSPN. This paper focuses on the practical aspects of the approach by showing its application to a non trivial modeling example related to a hypermedia system whose properties are predicted by exhaustive verification. Finally, conclusions are drawn with an indication of on-going and future work.**

Keywords-Model-based prediction; multimedia/ hypermedia systems; time stream Petri nets; synchronization consistency; model checking; UPPAAL.

I. INTRODUCTION

An interactive multimedia document (IMD) or hypermedia document, concerns the coordinated presentation of different types of information (audio, video, images, text, etc.) possibly associated with user interactions. The quality of the presentation of an IMD depends on the fulfillment of the temporal synchronization constraints, which are associated with the component media objects. Different approaches are described in the literature to address IMD specification and verification/validation of the synchronization consistency. Although an informal/intuitive or general purpose authoring approach can be preferable for editing an IMD, the informal specification must then the converted into some formal notation and associated tools, which can support the necessary verification activities. As an example, the SMIL language [3, 16] can be used as an authoring language. In the approach developed, e.g., in [15], from a SMIL specification some intermediate data structures are firstly generated; this makes it possible to translate the specification into the terms of process algebra of RT-LOTOS, which permits a formal analysis of synchronization consistency.

In this work, a methodology based on Petri nets [11, 13] is proposed for the specification and analysis of IMD. Petri nets have both an intuitive graphical notation and a rigorous mathematical representation for property checking. In particular, Time Stream Petri Nets (TSPN) [4, 10, 14] are chosen for the specification of complex hypermedia documents. TSPN associates temporal validity intervals to input arcs only, and a firing rule, selected in a rich set, to constrain transition firing. Although in the work described in [8-9] a TSPN specification or high level specification is translated into RT-LOTOS for verification purposes, the original contribution of this work is an exploitation of a translation of TSPN into UPPAAL [1], assisted by a library of reusable timed automata [7], which opens model checking to TSPN-based hypermedia documents. The achieved UPPAAL translation was proved to be timed bisimilar to TSPN. TSPN offers great flexibility to the modeler of general time-dependent systems. In [7], it is shown how the formalism is also well suited for modeling and property checking of real-time embedded systems.

Section II introduces the TSPN formalism and its extension HTSPN (Hierarchical TSPN) more suited to hypermedia systems modeling. Flattening problems of HTSPN to TSPN are addressed in Section III with a hypermedia example. The UPPAAL library supporting TSPN is summarized in section IV. Analysis of the synchronization consistency of the hypermedia example is reported in Section V. Finally, conclusions are presented together with an indication of on-going and future work.

II. BASIC CONCEPTS OF TIME STREAM PETRI NETS

A TSPN is a tuple $(P, T, B, F, I_{nh}, M_0, IM, SYN, MA)$ where:

- P is a finite nonempty set of *places;*
- T is a finite nonempty set of *transitions;*
- B is the backward incidence function, $B: P \times T \rightarrow \mathbf{N}$, where $\mathbf{N}$ denotes the set of natural integers;
- F is the forward incidence function, $F: P \times T \rightarrow \mathbf{N}$;
- 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, $M_0: P \rightarrow \mathbf{N}$, 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. $IM: A \rightarrow \mathbf{Q}^+ \times (\mathbf{Q}^+ \cup \{\infty\})$, where $\mathbf{Q}^+$ represents the set of nonnegative rational values, $A = \{a = (p,t) \in P \times T \mid B(p,t) \neq 0 \vee (p,t) \in I_{nh}\}$ 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 function, which associates each transition with a firing rule: $SYN: T \rightarrow \{And, Weak-And, Or, Strong-Or, Master, Or-Master, And-Master, Strong-Master, Weak-Master, Pure-And\}$;
- MA is a function that associates a master arc to each transition whose firing rule requires it, $MA: T_m \rightarrow A$, where:

$T_m = \{t \in T \mid SYN(t) \in \{Master, Or-Master, And-Master, Strong-Master, Weak-Master\}\}$.

The marking of a TSPN is a function $M: P \to \mathbf{N}$, which associates each place with a number of tokens. An arc $a = (p,t) \in A$ is enabled by a marking M iff: $a \in I_{nh} \Rightarrow M(p) = 0 \wedge a \notin I_{nh} \Rightarrow M(p) \geq B(p,t)$. 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 preset. The set of output places of a transition is its postset.

The state of a TSPN model is a pair (M,I) where M is a marking and I is a mapping that associates each arc enabled in M with a dynamic temporal validity interval, i.e., $I: enArc(M) \to R^+ \times (R^+ \cup \{\infty\})$, $I(a) = [x(a), y(a)]$. The initial state of a TSPN is given by (M_0, I_0), where $\forall a \in enArc(M_0)\ \ I(a) = IM(a)$. A transition t_f is said to be *fireable* at relative time θ from state (M,I) if t_f is enabled by M and $low(t_f) \leq \theta \leq \min_{t \in enabled(M)} up(t)$. For any transition t enabled by M, its dynamic time interval $[low(t), up(t)]$ is determined as follows:

$$\begin{cases} \left[\max_{a\in A(t)}\{x(a)\}, \max\left\{\max_{a\in A(t)}\{x(a)\}, \min_{a\in A(t)}\{y(a)\}\right\}\right] & SYN(t)=And \\ \left[\max_{a\in A(t)}\{x(a)\}, \max_{a\in A(t)}\{y(a)\}\right] & SYN(t)=Weak-And \\ \left[\max_{a\in A(t)}\{x(a)\}, \max\left\{\max_{a\in A(t)}\{x(a)\}, y(a_m)\right\}\right] & SYN(t)=And-Master \\ \left[\max_{a\in A(t)}\{x(a)\}, \min_{a\in A(t)}\{y(a)\}\right] & SYN(t)=Pure-And \\ \left[\min_{a\in A(t)}\{x(a)\}, \max_{a\in A(t)}\{y(a)\}\right] & SYN(t)=Or \\ \left[\min_{a\in A(t)}\{x(a)\}, \min_{a\in A(t)}\{y(a)\}\right] & SYN(t)=Strong-Or \\ \left[\min_{a\in A(t)}\{x(a)\}, y(a_m)\right] & SYN(t)=Or-Master \\ \left[x(a_m), y(a_m)\right] & SYN(t)=Master \\ \left[x(a_m), \max\left\{\min_{a\in A(t)}\{y(a)\}, x(a_m)\right\}\right] & SYN(t)=Strong-Master \\ \left[x(a_m), \max_{a\in A(t)}\{y(a)\}\right] & SYN(t)=Weak-Master \end{cases}$$

where $a_m = MA(t)$ if $t \in T_m$. Let t be a transition fireable from state $S = (M,I)$. The state $S' = (M',I')$ reached by firing t at relative time θ, is determined as follows:

- $\forall p \in P\ \ M'(p) = M(p) - B(t,p) + F(p,t)$
- $\forall a \in enArc(M')$

$$\begin{cases} I'(a) = IM(a) & \text{if } a \notin enArc(M) \vee \\ & a \notin enArc(\tilde{M}) \vee a \in A(t) \\ I'(a) = \left[\max\{x(a) - \theta, 0\}, y(a) - \theta\right] & \text{otherwise} \end{cases}$$

where $\forall p \in P\ \ \tilde{M}(p) = M(p) - B(p,t)$, i.e., $\tilde{M}$ is the intermediate marking resulting from the withdrawal sub-phase. An enabled arc is *violated* if the upper bound of its dynamic temporal validity interval is negative.

A transition t' enabled in M can lose its enabling during the atomic firing process of t either in the intermediate marking $\tilde{M}$ 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 $\tilde{M}$ but it is enabled in M'. It is worth noting that 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 (as in classical Time Petri Nets [12], which associates a static timing interval to transitions) on its firing. Of course, arc violations can determine transition violation if no valid timing interval is possible for the transition. As a consequence, TPSN use a *weak synchronization model* for arcs but a *strong synchronization model* for *transitions*.

The mathematical definition of transition dynamic firing interval highlights that, in general, a synchronization rule (see also Figure 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 the 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 the associated transition
(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.

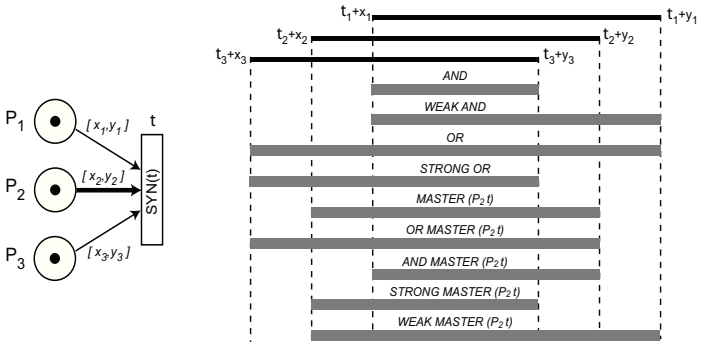

Figure 1. Basic TSPN synchronization rules

As an example of transitions, which can or cannot be violated, differences between the *And* and *Pure–And* firing rules can be pointed out. If input arcs temporal intervals overlap, the behaviour of *And* and *Pure-And* coincide: 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 (input arc) reaches its lower bound. In this situation, the *And* firing rule permits *one* single synchronization point at 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.

 ISBN: 978-1-61208-234-9

Rules *Master* and *Or–Master* can be violated when the master arc is violated. An intriguing case concerns the *Strong–Master* rule. If at the enabling time of the transition a non master arc is already violated, there exists one single synchronization point at the lower bound of the dynamic interval of the master arc. A single point of synchronization at current time is permitted when the master arc is violated at the enabling time of the transition. In the case of an overlapping scenario of the timing intervals, a firing interval for the transition exists, which starts with the lower bound of the master arc (possibly updated to current time if it was passed) and ends with the earliest upper bound. It can easily be verified that rules *Weak–Master*, *And–Master*, *Or, Strong-Or* and *Weak-And* can never be violated.

As it is usual in Time Petri Nets [2], a TSPN multiple enabled transition is assumed to fire its enablings one at a time (*single server semantics*). After its own firing, a transition t, which is still enabled is regarded as a newly enabled one.

III. MODELING A HYPERMEDIA SYSTEM BY TSPN

In order to widen the modeling capabilities of TSPN as an authoring formal tool, in [8], it was proposed HTSPN (Hierarchical TSPN). The concept was introduced of an *abstract place* shown graphically as a dashed circle (see Figure 2). A subnet can be defined within an abstract place, which typically has an input place and a final one. The subnet in turn can have further abstract places and so forth recursively.

Abstract places enable incremental modeling by deferring to a later time the definition of the internal details of the subnet. However, an abstract place must be both *structurally* and *temporally equivalent* to the internal subnet. Structural equivalence means that the subnet could functionally replace the abstract place. Temporal equivalence means that the expected temporal behavior of the subnet must fulfill the temporal constraints expressed at the abstract place (requirement) level.

To favor the authoring of complex multimedia/hypermedia systems, the use of HTSPN can be organized into three synchronization layers (see also Figure 2). At the *link layer* the hypermedia system is designed according to the user-point of view, i.e., conceptually in a similar way to an hypertext. Here, the user can choose a link to interrupt and put forward a given presentation or can require to rewind the presentation from its beginning etc. At the *composite layer*, the hypermedia system is specified through the composition of multiple media, which are operated according to given temporal constraints. Finally, at the *atomic layer,* details concerning the playing/rendering of a multimedia scenario are furnished.

Figure 2 portrays an example hypermedia model adapted from [8], devoted to the presentation of a commercial product. An initial text is presented to the user, which invites to start the presentation by clicking on a Start button. Following a start (modeled by next1 in Figure 2 and the firing of the first Master transition), a token is generated in the place L_2 and in the abstract place Information.

Token in L_2 enables both the possibility for the user to request the Again link, which asks to interrupt current presentation and to restart it from the beginning, or to conclude the presentation by invoking the Next link (next2 arc). The following constraints exist: (a) Again cannot be issued before at least 65 time units (tu) are elapsed from the moment the token was put in L_2; (b) the Next link can only be requested at the end of multimedia presentation. The multimedia presentation is assumed to last in 150 tu. Moreover, 70 tu are allowed, after termination of multimedia presentation, during which the user can ask the Next link for concluding the presentation (see next2 arc). If 70 tu elapses, the Next link is automatically invoked (see the Weak-And transition, which feeds the End place).

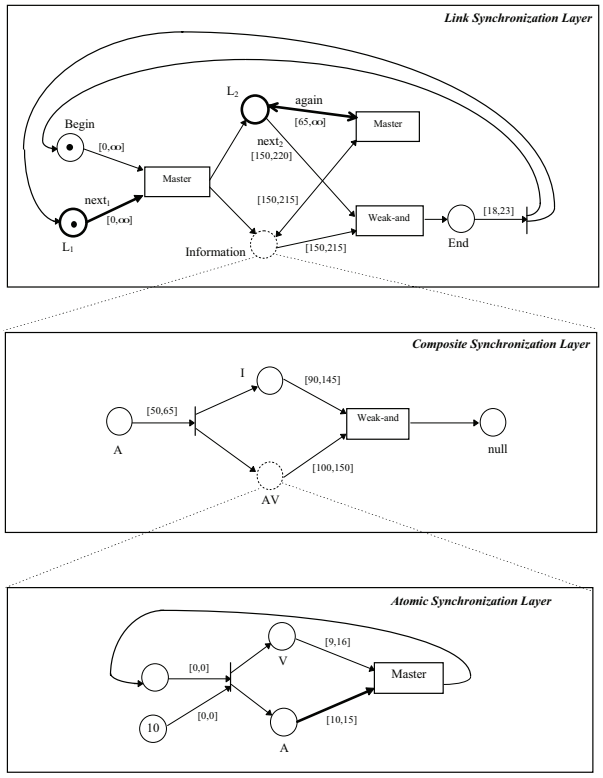

Figure 2. An HTSPN hypermedia model (from [8])

Token in Information abstract place begins the actual presentation. First an audio is played (see token in the A place in the composite layer in Figure 2), which in the worst case has a duration of 65 tu. The system forbids the user to ask the Again link if the audio content is still in progress. At the end of the audio, a token is generated in the place I of the subnet at the composite layer and in the AV abstract place. As a consequence an image is rendered with duration [90,145] tu. Token in AV starts an audio/video scenario detailed in the atomic layer. This multimedia scenario is composed of 10 audio/video objects. A video object (token in the V place) is rendered with a duration of [9,16] tu. A corresponding audio object (token in the A place) has a rendering timing constraint in [10,15] tu. Since the audio is the most important media, the master rule is used for synchronization and the input arc of the audio is defined as the master arc. As one can see from the atomic layer, inter-media synchronization (which affects skew and then *lip-synch*) was introduced

 ISBN: 978-1-61208-234-9

just after each presentation of a couple of corresponding audio and video objects. The goal was to minimize the audio/video skew by keeping audio/video synchronized to the largest extent.

The multimedia scenario terminates when all the 10 tokens are consumed. During the presentation of the multimedia scenario, the user can select the link Again, which requires the hypermedia system to interrupt its current operations and to restart from its beginning.

Temporal intervals on the arcs exiting from Information and AV abstract places capture system requirements. The requirement of finishing AV states that the multimedia scenario should terminate within [100,150] tu, which is intuitive by considering that audio objects are most important than video objects and that each audio requires [10,15] tu to be presented. The requirement of conclusion of Information states that the underlying layers must terminate within [150,215] tu (inclusive of the duration of the audio in the A place of the composite layer).

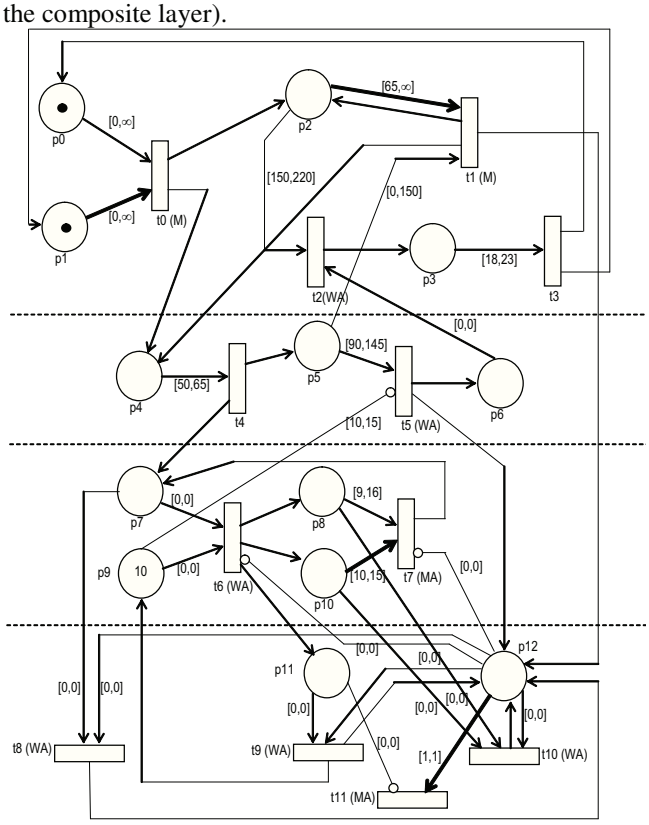

Figure 3. A flattened version of the hypermedia TSPN model

Following a termination of the multimedia scenario, a token is generated in the End place, which models some concluding remarks to the user. Such remarks have a duration in [18,23] tu. When also the remarks are finished, the system comes back to its home marking.

The conceptual model in Figure 2 specifies only hypermedia author requirements. It tacitly delegates the implementation level to correctly restart the Petri net to its home marking following the natural conclusion of the presentation or after each Again request.

In order to clarify problems that arise when an HTSPN hypermedia model like that portrayed in Figure 2 is put in concrete terms, Figure 3 shows a flattened version of the model. The unfolded model highlights dependencies between adjacent layers and, most importantly, solves problems of "dead tokens", which can be left in the model after an Again request. In addition, the flattened model is also in charge of re-installing the initial marking in all the cases it is necessary. A fourth layer, which can be named *clean-up layer*, was introduced with a minimal subnet, which eliminates no longer needed tokens and reconstitutes the model home marking.

For simplicity, in the flattened model a more uniform naming schema was followed for identifying both places and transitions. The clean-up subnet is composed by places p11 and p12 and transitions t8, t9, t10 and t11. At each termination of the multimedia scenario, or following an Again request, a token is deposited in p12, which supervises the model clean-up. Clean-up operations consumes 0 time. The token in p12 is eventually eliminated by a firing of transition t11, which uses the Master firing rule. One time unit is used for ensuring the end of the clean-up operation. Place p11 is the complementary one of the place p9. After each firing of t6 transition, a token is consumed from p9, and a new token is generated in p11. As a consequence, during the clean-up operations, t11 will fire multiple times so as to re-install the 10 initial tokens in p9. It should be noted that an Again request can occur during the presentation of a video/audio couple (one token in p8, one token in p10) or just before the beginning of the next multimedia couple of objects. Therefore, the clean-up subnet will remove no longer useful tokens in p7 or in p8 and p10.

A few inhibitor arcs were introduced in the flattened model in Figure 3. Arc p9t5 ensures the multimedia scenario naturally ends when all the 10 tokens in p9 are consumed. Arcs p12t6, p12t7 forbid prosecution of the multimedia scenario as soon as an interruption request (i.e., Again) is sensed. Finally, inhibitor arc p11t11 ensures that the p12 clean-up token can be destroyed after the home marking of p9 was re-established.

IV. AN UPPAAL TA CATALOG SUPPORTING TSPN

A structural translation was defined in [7], which enables a TSPN model to be equivalently transformed in a network of UPPAAL timed automata [1]. The translation associates a timed automaton with each transition with a given firing rule and a certain number of input arcs. Clocks are associated with input arcs only, but they are consulted by the transition according to the chosen synchronization rule. A transition template automaton is parameterized with its unique ID, and clock and bounds of its input arcs. The realization owes to latest UPPAAL version (4.0.13), which makes it possible to introduce global/local data structures and functions (in C-like syntax), which contribute to a compact definition of template processes without impairing the efficiency of state graph exploration.

Global constants A, P and T, respectively hold the number of input arcs, the number of places and the number of transitions in the model. Constants PRE and POST denote respectively the maximal cardinality of a transition preset or postset. Constant INF denotes infinity. Topology of a TSPN model is memorized into *constant*

 ISBN: 978-1-61208-234-9

matrices Backward B:TxPRE→ArcInfo, Forward F:TxPOST→F, InpurArcs AI:A→InputArcInfo where ArcInfo and InputArcInfo are the following structures:

```
typedef struct{
    int[-1,P-1] index; //place id, -1 for a non existing arc
    int weight; //input/output arc weight
} ArcInfo;
typedef struct{
    int[0,T-1] tid; //transition id
    int[0,PRE-1] pid; //preset id
} InputArcInfo;
```

Dynamic status information of a model is stored in the marking vector M, the enabled input arc vector EA, the input arc clock vector x and the current fired transition variable TID:

```
int[0,K] M[P]={...}; //for a K-bounded a model
bool EA[A]; //initialized to all false
clock x[A]; //one clock per input arc
int[-1,T-1] TID; //transition fired ID, -1 denotes no transition
```

Transition enabling and firing are assisted by the global functions bool enabled(const int ID), void withdraw(const int ID), void deposit(const int ID), which receive the unique transition ID. Input arc statuses and associated clock variables reset are responsibility of the global function void updateArcs(const bool w), which receives a boolean parameter indicating if the update is requested by a withdraw or a deposit operation. When an input arc *a* switches from the disabled to the enabled status or following the firing of its own transition, is still enabled, its status is set in the EA[a] and its clock x[a] is reset so as to start measuring the elapsed time since the instant of arc enabling. Similarly, when *a* is found disabled its status is reset in EA[a] and its clock x[a] is reset provided the function updateArcs() is invoked during a deposit or *a* refers to a transition t different from the current fired transition held in the global TID variable. This provision allows one to check, during verification, the clock value x[a] when a fired transition is in the intermediate withdraw phase.

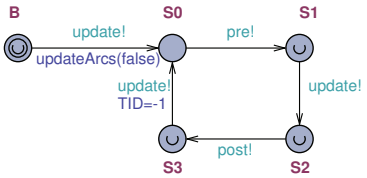

Figure 4. The Supervisor automaton

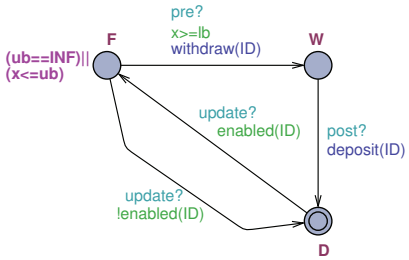

Figure 5. The Transition automaton

A TSPN model with *N* transitions is mapped onto an UPPAAL network of *N*+1 automata where each automaton corresponds to a distinct TSPN transition of the source model. The additional automaton is a supervisor [6] (see also Figure 4) whose responsibility is to allow TSPN transitions to complete their firing one at a time, by stepping through the atomic phases of transition firing.

A fundamental template is Transition (see Figure 5), which has one single input arc and can be used with any firing rule. A Transition instance receives the arc clock x and bounds lb and ub of the arc temporal validity interval.

A Transition automaton starts in the D (disabled) location. As soon as it finds itself enabled, it moves to the F (Firing) location, where it waits for the lower bound lb to be reached. Firing can be completed at any time the clock x is greater than lb but lower than or equal to ub (as stated by F invariant) if ub is finite. Would ub be infinity, the transition can stay in F an arbitrary amount of time. While in F, the transition can move immediately to D if the firing of another transition disables (for a conflict) this transition. Firing completion is mediated by the intervention of the Supervisor, using the two unicast channels pre and post and the broadcast channel update. The first phase of firing completion is for the transition to move from F to W (withdraw) location, under a pre synchronization. In the case multiple transitions are ready to complete their firing, one transition is chosen non deterministically. From W the transition eventually completes its firing by a second synchronization with the Supervisor through the post channel. Before this, the supervisor has to ask *all* the transitions to check their status following the withdraw of current firing transition. This important check is based on the broadcast channel update. A variable number of transitions (even no one transition) can possibly synchronize (update?) with the supervisor. Following an update synchronization, a transition can become enabled or being previously enabled (in the F location) it can become disabled. Urgent locations in the Supervisor ensure the firing process is terminated without passage of time. As one can see from Figure 4, the supervisor cycle includes two update broadcast synchronizations: after token withdrawl and after token deposit.

Initially, the Supervisor sends a first update synchronization so as to allow transitions that are enabled in the initial marking to switch from D to F location. The function updateArcs() is invoked after each change in the model marking, and thus after a withdrawl (within function withdraw()) or a deposit phase (within function deposit()). Initially, in order to permit input arcs to check their status, updateArcs() is invoked by the Supervisor along with the first update! synchronization.

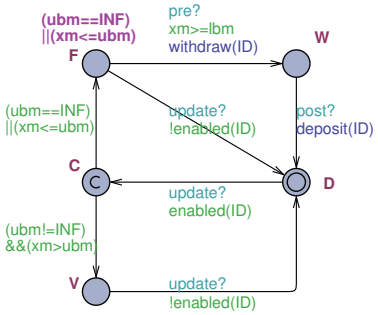

Figure 6. MasterTransition automaton

 ISBN: 978-1-61208-234-9

Transition template automata have names that mirror the adopted firing rule and the handled number of input arcs. In Figure 6 and Figure 7, the MasterTransition and WeakAnd2Transition templates are shown.

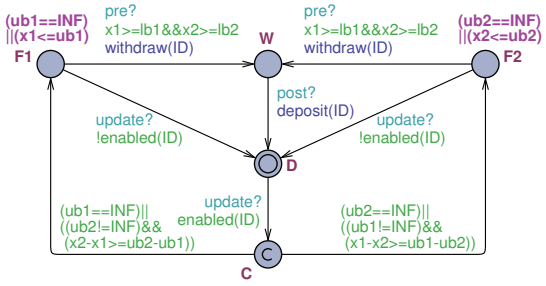

Figure 7. WeakAnd2Transition automaton

The automata in Figure 6 and Figure 7 illustrate some design principles, which were followed during library development. Each template is a decoration of the basic Transition template, according to the firing rule. A decision point C (committed location) is introduced where the automaton chooses to move to a firing location or to a V (Violation) location.

The MasterTransition is unique, whatever is the number of input arcs. It receives as parameters the transition ID and the clock xm and bounds, lbm and ubm, of the master arc. From C the automaton moves to V if the master arc is violated at the enabling time of the transition. Otherwise, it reaches the F location. It is worth to recall that from both urgent and committed locations an automaton has to exit immediately without passage of time. Committed locations, though, have greater priority than urgent locations.

In Figure 7, one can see two firing locations: F1, reached when the upper bound of first interval is found to be the maximum among the two intervals at the enabling time of transition, and F2 where the upper bound of the second interval is the maximum. From the guards of edges linking F1 or F2 to W, it is possible to see the *and* condition: only when both intervals are temporally ready, the transition can complete its firing.

Generally speaking, at the enabling time of a transition, for each input arc the two quantities can be evaluated: $lb_i - x_i$, which is the *time to start* of the relevant interval, and $ub_i - x_i$, which is the *time to finish* of the arc. In location C of Figure 7, if ub1−x1>=ub2−x2 it means that the maximum upper bound comes from the first arc. The relationship can be rewritten as x2−x1>=ub2−ub1 and so forth.

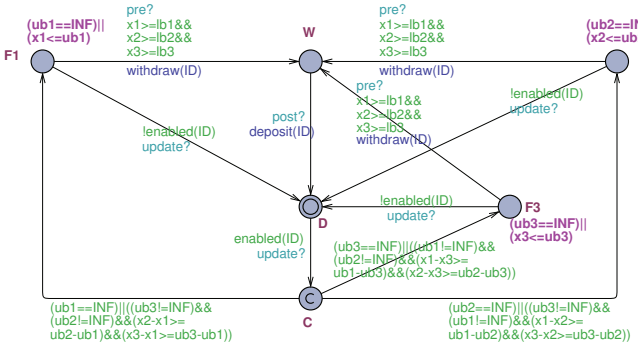

Figure 8. The WeakAnd3Transition automaton

Automata for two input arcs can easily be adapted to work with a greater number of input arcs. For example, Figure 8 shows the WeakAnd3Transition automaton. For details about all the other template automata of the developed UPPAAL library, the reader is referred to [7].

Figure 9 shows the system declaration section of the UPPAAL model corresponding to the translated flattened version of TSPN hypermedia example.

```
// Place template instantiations here.
t0=MasterTransition(T0, x[P1T0], 0, INF);
t1=MasterTransition(T1, x[P2T1], 65, INF);
t2=WeakAnd2Transition(T2, x[P6T2], 0, 0, x[P2T2], 150, 220);
t3=Transition(T3, x[P3T3], 18, 23);
t4=Transition(T4, x[P4T4], 50, 65);
t5=WeakAnd2Transition(T5, x[P9T5], 10, 15, x[P5T5], 90, 145);
t6=WeakAnd3Transition(T6, x[P9T6], 0, 0, x[P7T6], 0, 0, x[P12T6], 0, 0);
t7=MasterTransition(T7, x[P10T7], 10, 15);
t8=WeakAnd2Transition(T8, x[P7T8], 0, 0, x[P12T8], 0, 0);
t9=WeakAnd2Transition(T9, x[P11T9], 0, 0, x[P12T9], 0, 0);
t10=WeakAnd3Transition(T10, x[P10T10], 0, 0, x[P8T10],0,0,x[P12T10],0,0);
t11=MasterTransition(T11, x[P12T11], 1, 1);
// List one or more processes to be composed into a system.
system Supervisor, t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11;
```

Figure 9. System configuration of the hypermedia model in UPPAAL

V. MODEL CHECKING THE HYPERMEDIA MODEL

After translation in UPPAAL of the flattened version of TSPN model, it was possible to check its consistency synchronization. Table 1 summarizes the queries that were issued to the UPPAAL verifier for exhaustive property checking, and the gathered answers.

The actual declaration of the marking vector M was:
int[0,10] M[P]={1,1,0,0,0,0,0,0,0,10,0,0,0};

Since no out-of-range assignment was signaled by UPPAAL during analysis, it was concluded that effectively the TSPN model is 10−bounded as expected.

TABLE 1. PROPERTY CHECKING OF THE HYPERMEDIA MODEL

	Query	Answer
(1)	A[] !deadlock	Property is satisfied
(2)	E<> t3.W	Property is satisfied
(3)	t3.W-->(M[0]==1 && M[1]==1 && M[4]==0 && M[5]==0 && M[6]==0 && M[7]==0 && M[8]==0 && M[9]==10 && M[10]==0 && M[11]==0 && M[12]==0)	Property is satisfied
(4)	t1.W-->(M[9]==10 && M[7]==0 && M[8]==0 && M[10]==0)	Property is satisfied
(5)	t0.W --> t3.W	Property is not satisfied
(6)	t2.W --> t3.W	Property is satisfied
(7)	A[] t5.W imply y>=150 && y<=215	Property is satisfied
(8)	A[] t7.W imply z<=150	Property is satisfied
(9)	A[] t1.W imply x[P2T1]>=65	Property is satisfied
(10)	E<> t2.W && x[P2T2]>220	Property is not satisfied

A first concern (query (1)) was checking absence of deadlocks in all the states of the model state graph (a *safety* property). Operator A[] verifies that !deadlock is invariantly true in all the states of the model. Query (2) (*existential*) asks if there is at least one state of the state graph where transition t3, which concludes the whole presentation, fires (the automaton is found in the W location). Being known that t3 actually can fires, query

 ISBN: 978-1-61208-234-9

(3), based on the *leads-to* operator -->, checks if it is always true that starting from a state in which t3 fires, it always follows a state in which the initial marking of the model is reinstalled. Query (4) asks, similarly, that starting from a state in which t1 fires, i.e., the Again link was clicked, it always follows a state in which the atomic layer network, which plays the multimedia scenario (see places p7, p8, p9 and p10) reaches the home marking in which the marking of p9 is 10 (the number of couple of media objects to be played) and the rest of the subnet is reset. This property in turn ensures that at any time the link Again is requested, the subnet correctly starts from its home marking. Queries (5) and (6) check *liveness* properties. In particular, query (5) asks the verifier if starting from a firing of t0 (which begins the presentation) it always follows a firing of t3 (which concludes the entire presentation). Obviously, this query has a negative response because once started the presentation can be interrupted by an Again request possibly an infinity number of times. Of course, query (6) is always true: after a conclusion of the multimedia scenario, it always follows the concluding remark is rendered. Queries (7) and (8) are examples of *bounded liveness* property checking. Toward this, two extra clocks z and y (*decoration clocks*) were added to the model. Clock y is reset at each firing of t0 or t1, which starts a new presentation, whereas clock z is reset at each firing of t4, which begins the multimedia scenario of the atomic subnet. All of this was achieved by adding a few instructions to the deposit() function. Query (7) asks if it is always true that a state in which t5 fires implies that a number of time units between 150 and 225 (including the audio presentation in place p4, which lasts in [50,65] time units) elapse (an expectation). Query (8) checks if it is always true that a state in which t7 fires implies that clock z is less than or equal to 150 time units. Query (9) checks if invariantly, i.e., in all states in which t1 fires (the Again link was requested), the arc clock x[P2T1] has always a value not less than 65. This property guarantees that, as in the requirements of the hypermedia model, the Again link cannot be issued before at least 65 time units are elapsed. Finally, query (10) verifies that in no case the arc clock x[P2T2] can be greater than 220 at the time in which t2 fires. This property ensures that, following a termination of the multimedia scenario, the Next link at most after 70 time units, that is after 220 time units, is automatically invoked.

Due to answers collected in Table 1, the TSPN hypermedia model was found correct functionally and temporally. The experimental work was carried out on a Win7, Intel Core i3, 4GB, 2.13 GHz. To give an idea of the efficiency of the achieved implementation, any query in Table 1 ends in about 5 seconds.

VI. Conclusion And Future Work

This paper proposes a methodology for modeling and analysis of multimedia/hypermedia documents, which is based on the Time Stream Petri Net formalism [10]. A TSPN model is translated into UPPAAL with the help of a developed reusable library of timed automata [7]. All of this enables synchronization consistency and temporal properties of the multimedia document to be verified through model checking.

Prosecution of the work aims to:

- Automating the translation from TSPN to UPPAAL, by completing an extension of the TPN/Designer toolbox [5] so as to graphically drawing a TSPN model and then generating the corresponding XML UPPAAL code.
- Extending the approach so as to consider some high level or general purpose authoring language like SMIL and converting an initial specification of an interactive multimedia document into the terms of TSPN for subsequent thoroughly exhaustive verification.

Acknowledgments

Authors are grateful to Christian Nigro for his contribution in the development/analysis of the hypermedia modeling example.

References

[1] G. Behrmann, A. David, and K.G. Larsen, “A tutorial on UPPAAL”, in *Formal Methods for the design of real-time systems, LNCS 3185*, Springer, pp. 200-236, 2004.

[2] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using Time Petri Nets”, *IEEE Trans. on Soft. Eng.*, **17**(3):259–273, 1991.

[3] S. Bouyakoub and A. Belkhir, “Towards a formal approach for modeling SMIL documents”, *IJSSST*, **10**(2):1-12, 2009.

[4] M. Boyer and M. Diaz, “Non equivalence between time Petri nets and time stream Petri nets”, in *Proc. of 8th Int. Workshop on Petri Nets and Performance Models*, pp. 198–207, 1999.

[5] L. Carullo, A. Furfaro, L. Nigro, and F. Pupo, “Modeling and simulation of complex systems using TPN DESIGNER”, *Simulation Modeling Practice and Theory*, **11**, pp. 503-532, 2003.

[6] F. Cassez and O. H. Roux, “Structural translation from time Petri nets to timed automata”, *J. of Systems and Software*, vol. 79, pp. 1456–1468, 2006.

[7] F. Cicirelli, A. Furfaro, and L. Nigro, “Model checking time-dependent system specifications using time stream Petri nets and UPPAAL”, *Applied Mathematics and Computation*, **218**(16):8160-8186, 2012.

[8] J.-P. Courtiat, Diaz M., R.C. De Olieira, and P. Senac, “Formal models for the description of timed behaviors of multimedia and hypermedia distributed systems”, *Computer Communications*, vol. 19, pp. 1134-1150, 1996.

[9] J.-P. Courtiat and R.C. De Olieira, “Proving temporal consistency in a new multimedia synchronization model”, in *Proc. ACM Multimedia,* 1996.

[10] M. Diaz and P. Sénac, “Time Stream Petri Nets, a model for multimedia streams synchronization”, in *Proc. of Multimedia Modeling (MMM’93)*, 1993.

[11] W. Liu and Y. Du, “Modeling multimedia synchronization using Petri nets”, *Inf. Technology Journal*, vol. **8**, pp. 1054-1058, 2009.

[12] P. Merlin and D.J. Farber, "Recoverability of communication protocols”, *IEEE Trans. Commun.*, **24**(9):1036-1043, 1976.

[13] T. Murata, “Petri nets: properties, analysis and applications”, in *Proc. of the IEEE*, **77**(4):541-580, 1989.

[14] P. Sénac, M. Diaz, A. Léger, and P. de Saqui-Sannes, "Modeling logical and temporal synchronization in hypermedia systems”, *IEEE J. on Selected Areas in Comm.*, **14**(1):84–103, 1996.

[15] P.N.M. Sampaio and J.-P. Courtiat, “An approach for the automatic generation of RT-LOTOS specifications from SMIL 2.0 documents”, *J. of Brazilian Computer Society*, **9**(3), 2004.

[16] SMIL on-line, http://www.w3.org/TR/smil20 [retrieved: September, 2012].

 ISBN: 978-1-61208-234-9