Application of Compositional Petri Nets and PN³–Tool to the Specification of Distributed Multimedia Objects

NIKOLAY ANISIMOV, ALEKSEY KOVALENKO

PAVEL POSTUPALSKY

Institute for Automation and Control Processes
of the Russian Academy of Sciences, 5 Radio street, Vladivostok 690041, Russia
E-mail: anisimov@iapu2.marine.su

and

SON VUONG

University of British Columbia, Vancouver, Canada
E-mail: vuong@cs.ubc.ca

This paper addresses the problems of specifying synchronization requirements for real-world size multimedia presentations. In particular, we propose a compositional Petri net model, called CoPN, for multimedia synchronization specifications. The salient features of this model, including *macroplaces* and *PN entities*, are presented and the application of the the CoPN model to multimedia specification is discussed via a number of illustrative examples of intra- and inter-stream multimedia synchronizations. This compositional approach enables compact and readable specification of complex, large-scale specifications while preserving the fine granularity as well as supporting user interactions. Preliminary results on incorporating the time stream Petri net (TSPN) in the compositional method for handling time are discussed. The Petri net tool, called PN³-Tool, is also presented, which can be applied to verify the correctness of specifications of multimedia synchronizations in CoPN.

Keywords: multimedia system, inter-stream, intra-stream synchronization, user interaction, Petri nets, compositional Petri nets, time stream Petri nets, compositionality, modularuty

1 Introduction

An important problem in multimedia communication is the synchronization of both static (data, text) and dynamic (video, audio) objects. To some extent, this prob-

lem can be successfully solved with the aid of Petri net models (Reisig, 1985; Brauer *et al.*, 1987) such as Time Stream Petri Nets (TSPN) (Diaz and Senac, 1993; Sénac *et al.*, 1994) and itsextension for user interaction $TSPN_{ui}$ (Cooper, 1995), Object Composition Petri Nets (OCPN) (Little and Ghafoor, 1990), Extended Object Composition Petri Nets(XOCPN) (Woo *et al.*, 1994), Dynamic Timed Petri Nets (Prabhakaran and Raghavan, 1993). A comprehensive review of these models can be found in (Vuong *et al.*, 1995). However, a common problem with the existing specification methods lies in the complexity of the specifications for real-life multimedia application. A viable way to cope with this problem is to apply the notion of compositionality and modularity which was recently introduced in the CoPN model (Anisimov and Koutny, 1996; Best *et al.*, 1992). We should note existing work which addresses similar problems of structured specification, such as the one reported in (Sénac *et al.*, 1996) where the TSPN is extended into the hierarchical TSPN (HTSPN), which has features similar to macronets. Also, the work by Courtiat *et al.* incorporates features for structuring multimedia specification where the real-Time LOTOS is used as the underlining model. Thus, this model inherits both the advantages of existing LOTOS support tools as well as the potential disadvantage of the complexity of LOTOS as a specification language.

The main contribution of our paper rests on the application of the compositional approach developed in (Anisimov,1989; Anisimov, 1991; Anisimov and Koutny, 1996) for the specification of complex real-life multimedia systems. In this paper, we review the salient features of the CompositionalPetri net (CoPN) model: *macroplaces*, *Petri net entities* and *access points*; and discuss their applications in multimedia synchronization specification via several illustrative examples. In the next section, the notion of of macroplace is introduced and shown how it can simplify large multimedia specifications that involve user interactions. Section 3 defines Petri entity which can be nicely applied to the specification of intra- and inter-stream multimedia synchronization. In Section 4, we briefly describe a CoPN tool, called PN^3-Tool, which can be used to graphically generate CoPN specifications. To illustrate the viability of the CoPN model, a complete example of multimedia scenario specification is presented in Section 5, followed by some concluding remarks in Section 6. This paper is a revised and extended version of the conference paper (Anisimov *et al.*, 1997).

2   Petri nets with macroplaces

In recent work (Anisimov, 1991; Anisimov *et al.*, 1994; Anisimov and Koutny, 1996) the classic Petri Nets have been extended with the notion of *macroplace*. Such extended nets are called *macronets*. In this section we will show how the use of macroplaces and macronets improves the process of multimedia specification.

Macronet is a regular Petri net augmented with a set of macroplaces. First, we will introduce the *simple macroplace*. Informally, a macroplace is a set of places,

called *internal* places, one of which is marked as a *head place* of the macroplace. Macroplaces are connected to/from other (macro)places via transitions in the same way as simple (non-macro) places. So, a macroplace is considered to be marked if at least one of his internal places is marked. Adding the token to a macroplace is defined as putting a token into its internal head place. Removing the token from the macroplace leads to removing the token from one of the internal places of the macroplace. The transition of macronet is considered to be enabled if its each input (macro)place has at least one token. Then, firing of enabled transition involves removing token from each input (macro)place and adding token to each output (macro)place.

Intuitively, the above definition of macroplace makes sense only if its internal places form the state-machine net (SM-net), i.e. a Petri net in which each transition has exactly one incoming and one outcoming arc. A macroplace corresponding to a SM-net will be called *simple macroplace*.

Formally, a macronet is defined as a tuple $N = \langle S, T, F, M_0, Q \rangle$, where:

- $S = S^n \cup S^\mu$, where $S^n \cap S^\mu = \emptyset$. S^n is the set of simple (non-macro) places, and S^μ is the set of macroplaces;
- T is the set of transitions, such that $S \cap T = \emptyset$;
- $F \subseteq S \times T \cup T \times S$ is a flow relation;
- $M_0 : S^n \rightarrow \{0, 1, 2, \ldots\}$ is an initial marking;
- $Q : S^\mu \rightarrow \mathcal{P}(S^n) \times S^n$, such that $\forall s \in S^\mu$: $Q(s) = (S', s_0) \Rightarrow s_0 \in S'$. Q is a function that associates with each macroplace $s^\mu \in S^\mu$ a set of internal places S' and a head place s_0.

Graphically, a macroplace is depicted as a large circle surrounding the part of the net formed by its associated internal places. Incoming and outcoming arcs are attached to a macroplace in the same way as to an ordinary place.

Fig.1 shows an example of a macronet. This net has a macroplace s_7 with internal places $\{s_5, s_6\}$. Macroplace s_7 will be marked if either s_5 or s_6 has a token. Firing t_5 then will lead to putting a token into the head place s_5. Transition t_4 will be enabled if s_3 and s_7 are marked. In other words, t_4 can fire if either the set s_3, s_5 or s_3, s_6 have tokens.

The specification paradigm depicted in Fig.1 can be quite handy for designating a stream as the multimedia master stream among other streams. For example, part of the stream s_5, s_6 can be "disrupted" by the transition t_2 at any time.

Macronets also allow the specification of a complex system in a more compact and natural way than regular Petri nets. For instance, the macronet in Fig.2 models the repeated background stream (e.g. audio, video, etc.) which can be terminated by the master stream regardless the state (marking) this background stream may be in.

It should be noted that macronets are merely a compact representation of classic Petri nets of complex internal structure. It is relatively straightforward to transform
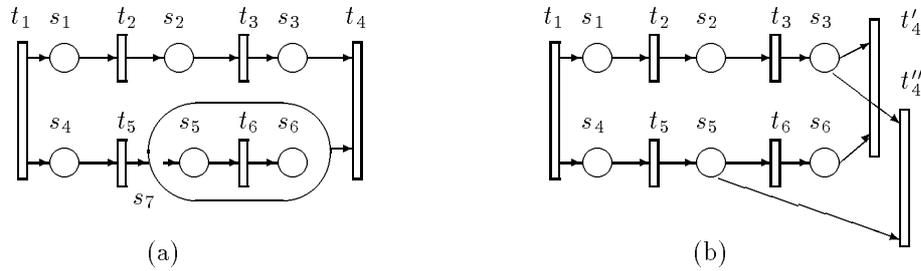
Fig. 1   (a) Macronet with simple macroplace (b) Equivalent regular Petri net
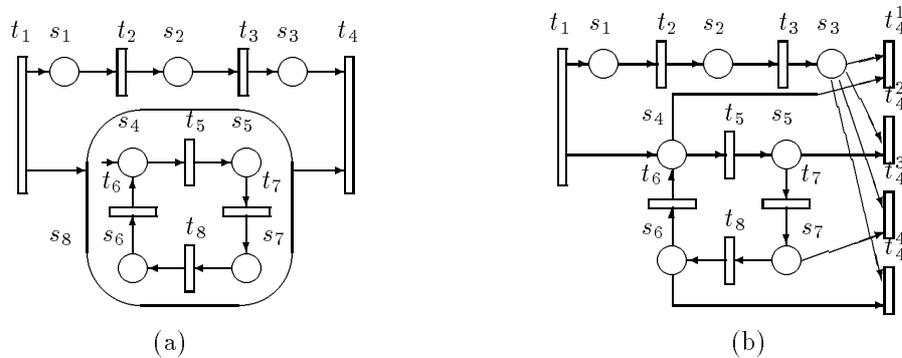

Fig. 2   Cyclic background specification.

a macronet to its equivalent regular Petri net. For example, the macronet shown in Fig.1(a) can be easily transformed into the Petri net shown in Fig.1(b). We can see that transition t_4 is split into two copies t'_4, t''_4, each corresponds to the possible marking of either s_5 or s_6.

So far, we have addressed only simple macroplaces whose internal places form a SM-net. We can define a general form of macroplace which consists of the parallel composition of SM fragments. In this case we have to designate a head place for each SM fragment. Respectively, putting (or removing) the token in a general macroplace leads to putting the token in each head place (or removing the token from each SM fragment). A formal definition of the general form of a macroplace can be found in (Anisimov, 1991; Anisimov and Kovalenko, 1995).

Using a general form of macroplace, we can define the operations of user inter-action such as **terminate**, **restart**, and **forward**. In Fig.3, executing transition $t_{terminate}$ causes a termination of all activities in the internal net corresponding the the macroplace, i.e. all internal tokens (from both upper and lower streams) are removed. Transition $t_{forward}$ has similar effect on the internal net. The difference between these two actions is that in the future $t_{forward}$ and t_{end} will together be

merged with the first transition of next stream whereas $t_{terminate}$ will be merged with a special termination transition. The execution of $t_{restart}$ will remove a token from $s_{7_{macro}}$, i.e. all tokens from the internal net, and then reinitialize all its head places, s_1 and s_4. The macronet Fig.3 can be transformed into a classic Petri net, but it will lead to an extremely messy specification, e.g. the transition $t_{restart}$ will be split into 16 copies. The above user interaction functions would be very useful for video-on-demand applications.
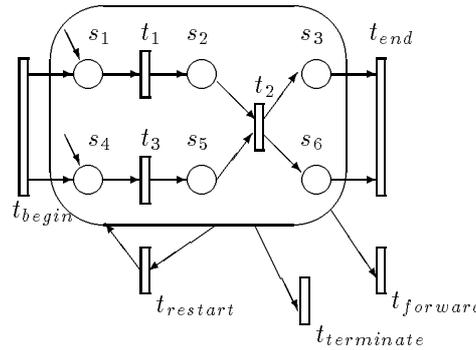

Fig. 3   User interaction: restart, terminate, forward.

Thus, we can see that macronets offer a way of multimedia synchronization specification with more descriptive power while maintaining the compactness of the specification. Using macronets it becomes possible to specify new types of constructs such as cycles as illustrated in Fig.2. Moreover, some user interaction operations such as restart, terminate and forward can be specified in a natural and compact way.

3   Petri net entity.

In this section, we will present the notions of *Petri net Entity* and *access points* used in CoPN to specify compositionality, and show via a number of examples how they can be nicely applied to the compositional and hierarchical specification of multimedia scenarios. The notions of Petri net entity and access points were first introduced in (Anisimov, 1989), and further developed in (Anisimov, 1991a; Anisimov, 1991c; Anisimov and Koutny, 1996).

Petri net entity is a generalization of well-known labeled Petri nets where instead of a single labelling function we allow several ones. These labeling functions are called *access points*. Formally, a Petri net entity (or entity for short) is defined as a tuple $E = \langle N, G \rangle$, where:

- $N = \langle S, T, F, M_0 \rangle$ is a Petri net;
- $G = \{g_1, g_2, \ldots, g_n\}$ is a set of access points $g_i = \langle id_i, Alph_i, \sigma_i \rangle$ where:
  - id_i is the identifier of the access point g_i;
  - $Alph_i$ is an alphabet (set of labels).
  - $\sigma_i : T \rightarrow Alph_i \cup \{\tau\}$ is a labeling function, where τ is a special symbol corresponding to an invisible action.

Graphically, an entity can be represented in two complementary ways: in a Petri net form and in a diagram form. In the first case, each transition of a classic Petri net is labeled by the set of expressions of the form $id_i : \xi;$. This means that the transition is visible as (interaction) ξ at access point (gate) names id_i. A transition may have several labels, i.e. it may be visible at different access points by different names. If a transition is labeled at g_i as τ it considers to be invisible at the access point g_i. Furthermore, labels such as $id_i : \tau$ will be omitted. In the diagram form, an entity is represented as a rectangle with small bold-face markers placed on its perimeter, where each marker represents an access point.

In Fig.4 the entity is shown in both forms. As shown in diagram form, the entity E has three access points L, R, and U, where L and R indicate the beginning and the end of the entity execution, respectively, whereas access point U is used for user interaction functions (in this case, only *advance*). In the Petri net form, transitions t_1 and t_3 are visible through L and R access points, respectively. The universal symbol — λ is used for development a composition on entities. The transitions t_3 and t_4 are visible through U as *advance*. Other transitions are not visible at all.
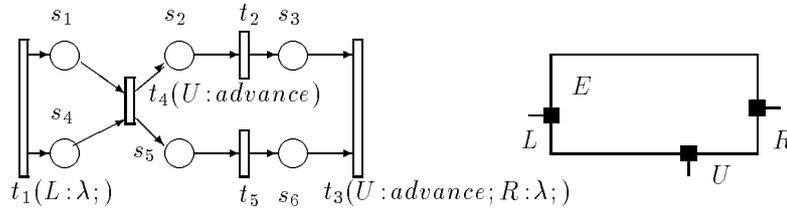

Fig. 4   Petri net entity example.

Composition of entities can be defined through their access points. The composition of two entities E_1 and E_2 through α and β produces a new entity $E_n = E_1 \; _\alpha\|_\beta \; E_2$. The operation involves disjoint union of corresponding nets N_1 and N_2, and then merging of transitions visible at α and β with the same names. The new set of access points will be $G = G_1 \cup G_2 \setminus \{\alpha, \beta\}$. The complete formal definition of the composition operation can be found in (Anisimov *et al.*, 1994). In the diagram form this operation is represented by connecting the appropriate access points of entities, as shown in Fig.5. For example, it is possible to specify the sequential behavior of the streams (i.e. intrastream synchronization) using the composition of

entities. Fig.5 shows the new entity $E_r = E_1$ $_R\|_L$ E_2 in the diagram form, composed by connecting the access points R and L, and in Petri net form by merging the transitions that are visible at R and L. In the simplest case, when only one transition in each entity is visible at R and L, respectively, these corresponding transitions will be merged. The result entity will inherit the set $\{L, R, U, T\}$ of access points while the connected R and L will be eliminated due to participation in the composition.
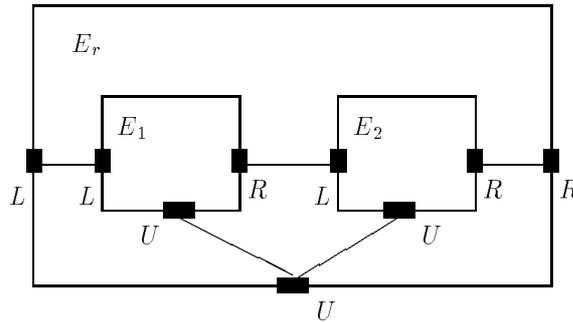

Fig. 5   Sequential composition of entities.

On the architectural level, a specification on CoPN consists of the set of hierarchical composed Petri net entities where the lowest level entities have the Petri net equivalent representation. Such level of specification can be view as a conceptual one. A formal semantic of such specification is defined as the result of performing all the entity compositions in the Petri net form. In general, the resulting Petri net can be unwieldily complex. However, it can either partially or fully processed for useful protocol engineering tasks such as verification, test generation and automatic implementation.

The main advantage of compositional specification, i.e. specification at the architectural level, is that it allows description of complex systems in terms of blocks and their interconnections which is more convenient for most practitioners then a single complex Petri net. But at the any phase in the protocol engineering process, the compositional specification can be converted to a Petri net for suitable processing.

Examples. We now present two simple examples to illustrate the applicability of the entity composition in the specification of interstream synchronization, as shown in Fig.6, and of a multimedia scenario with user interactions, as shown in Fig.7. Fig.6 shows two streams which are synchronized at every other transitions. The synchronization is shown by the same interaction label at the respective transitions. The Petri net corresponding to each entity is placed in an entity box. The Petri net corresponding to the composition of the two entities E_a and E_v through the access points A and V is shown on the right-hand side of Fig.6.
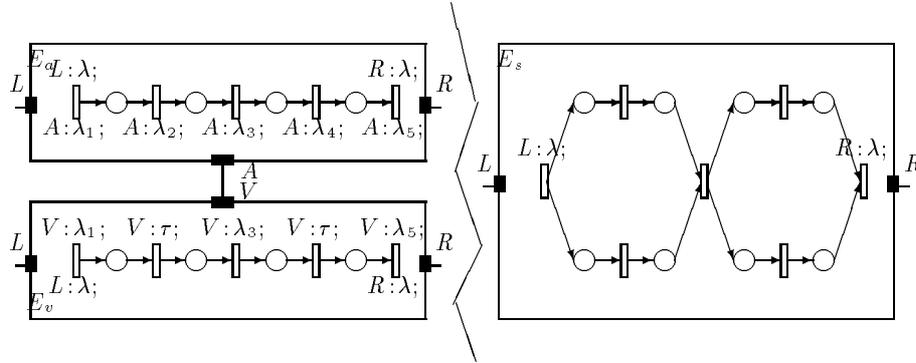
Fig. 6   Interstream synchronization.

User interaction operations such as **stop** and **resume** can be specified in a very elegant way via an extra controller entity, as shown in Fig.7. Each transition of the net which must be controlled (i.e. it may be affected by user interactions) must be visible at the access point by the respective user interaction label in order to be synchronized with the corresponding transition in the controller entity. In this example, every transition in the net is visible through U access point as ζ; they are synchronized with the (self-loop) transition with the same label ζ in the controller entity. After firing the **stop** transition in the controller entity, every (synchronized) transition in the net will be disabled until the **resume** transition is executed.

4   TSPN in entity composition.

We have so far considered Petri net entity composition to provide inter-stream synchronization without time constraints. However, since time constraints plays a key role in multimedia presentation, they must be taken into account during the composition of streams. In this section, we discuss the handling of time constraints by incorporating a basic time Petri net model with into the entity compositional approach. Among the various time Petri net models, we choose the Time Stream Petri Nets (TSPN) (Diaz and Senac, 1993) as the basic model, where the time supports for multimedia presentation are well elaborated. In TSPN, each input arc to a transition is associated with an interval $[\tau^{min}, \tau^{max}]$. Each transition is attributed with one of the firing disciplines: $\mathcal{R} = \{$and, weak-and, or, strong-or, master, or-master, and-master, weak-master, strong-master$\}$, as illustrated in Fig.8. Using arc intervals and transition firing discipline, we can calculate the transition enabling interval. For example, when the token reaches a place s_i, $i \in \{1, 2, 3\}$ at the moment τ_i^{tok} (Fig.8) the (s_i, t_i) timer for the input arc is started and the
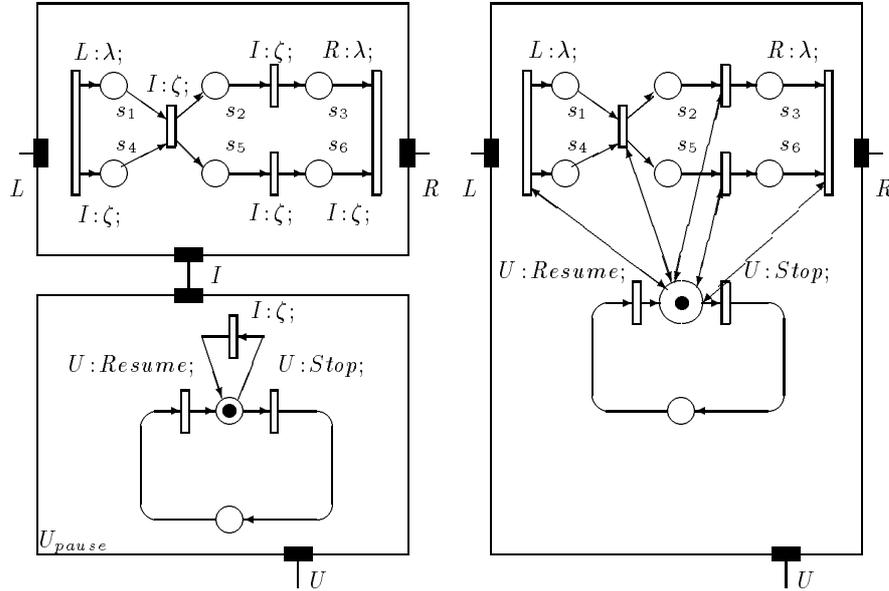
Fig. 7   Example with user interactions: stop and resume.

arc becomes enabled in the interval $[\tau_i^{tok} + \tau_i^{min}, \tau_i^{tok} + \tau_i^{max}]$. According to firing discipline **and**, the transition t_1 is enabled within the intersection of three intervals $\bigcap_{1 \le i \le 3}[\tau_i^{tok} + \tau_i^{min}, \tau_i^{tok} + \tau_i^{max}]$. For transition t_2 in Fig.8 with associated firing discipline **or**, the enabling interval is defined as the union of the arc's intervals. Other firing disciplines are discussed in details in (Diaz and Senac, 1993).

Let us now examine the time issues in context of the compositional approach. Suppose, we wish to perform an inter-stream synchronization of two streams that results in the merging of two transitions t_1 and t_2 to yield a new transition (t_1, t_2), as shown in Fig.8. The question arises is which firing discipline should be attributed to the new transition. Intuitively, we would expect the firing discipline in each stream to be preserved in the new transition. Moreover, we need an additional piece of information about the interplaying of the two streams. For example, we may want to synchronize t_1 as the master role. Thus, we can easily see that more than eight rules listed above will be needed.

We suggest a technique for the derivation of firing disciplines within the framework of our compositional approach. The main idea of our approach is as follows. Each firing discipline in $\mathcal{R}$ will be considered as a binary function which maps two intervals into a resulting interval. For instance, **and**(int_1, int_2) gives us a new interval which is the intersection of intervals int_1 and int_2. This resulting interval can be used as an argument in another function, e.g. **strong−or**(**and**$(int_1, int_2), int_3$). Thus it is possible to define firing disciplines of transitions as a superposition of
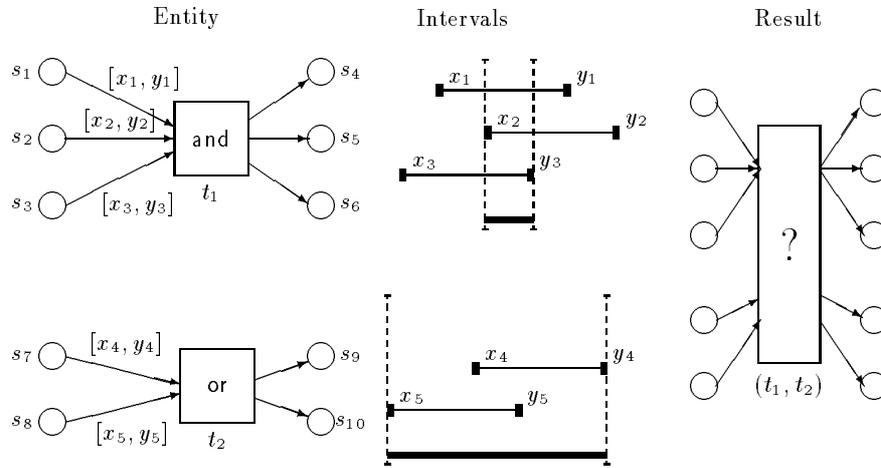
Fig. 8   A TSPN example.

functions whose arguments contain intervals of input arcs.

We note that in formulating the rules for composite firing disciplines, it is more convenient to manipulate with arcs than with its intervals. Thus, for each transition we assign a *firing discipline expression* (FDE).
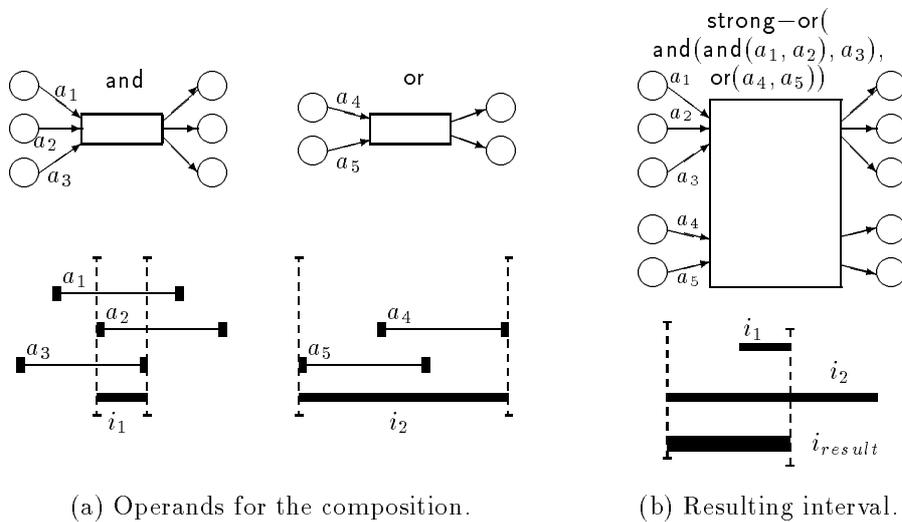

(a) Operands for the composition.   (b) Resulting interval.

Fig. 9   Calculating the resulting interval in a TSPN composition.

Let ex be an FDE, a be an input arc of transition t. Let *rule* vary over $\mathcal{R}^* = \{\mathsf{and}, \mathsf{weak-and}, \mathsf{or}, \mathsf{weak-master}_i, \mathsf{strong-master}_i\}$, $i \in \{1, 2\}$. We will write

$a \in ex(a \notin ex)$ if a is (not) participating in ex. Here, the index of a master participated discipline will indicate the master operand. We now define the FDE for a transition t recursively as follows:

(1) the empty string is a FDE;
(2) $rule(a_1, a_2)$ is a FDE if $a_1 \neq a_2$;
(3) $rule(ex, a)$ is a FDE if $a \notin ex$
(4) $rule(a, ex)$ is a FDE if $a \notin ex$
(5) $rule(ex_1, ex_2)$ is a FDE if there does not exist a such that $a \in ex_1$ and $a \in ex_2$.

The expressions $\mathsf{or}(a_1, a_2)$, $\mathsf{strong-or}(\mathsf{or}(a_1, a_2), a_3)$, $\mathsf{master}_1(\mathsf{weak-and}(a_1, a_2),$ $\mathsf{or}(a_3, a_4))$ are examples of FDEs. The expression $\mathsf{weak-and}(a_1, \mathsf{strong-master}_2(a_2,$ $a_1))$ is not a FDE because it has two instances of the arc a_1.

Since every arc has an appropriate temporal interval, the FDE allows for the unique calculations of the enabling transition interval. For example, the transition (t_1, t_2) is the result of the $\mathsf{strong-or}$ FDE of the arcs of the transitions t_1 and t_2, respectively, as illustrated in Fig.9(a). The resulting enabling transition interval is calculated as shown in Fig.9(b).

It should be noted that the graphical representation of transitions with firing discipline expressions is combersome because it clames identification of all incoming arks. For example, in the resulting transition (t_1, t_2) on the Fig.9(b) we have to label all five incoming arks $a_1, ..., a_5$. Less cumbersome representation of the transition is depicted in Fig.10(a) where the transition with FDE is substituted by the cascade of transitions corresponding to FDE*.
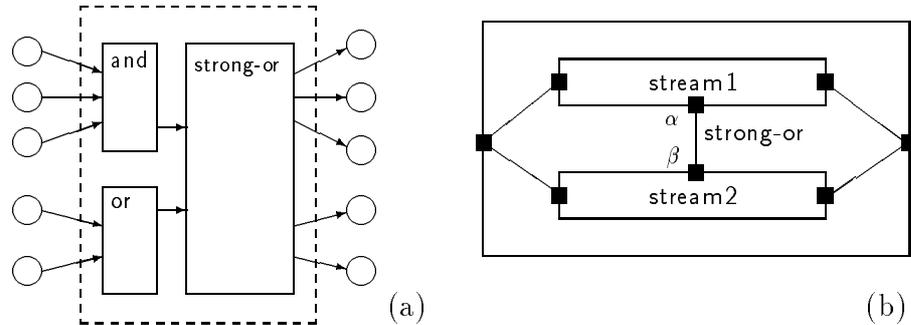

Fig. 10   Graphical representation of firing discipline expression

Information about the rule for an inter-stream synchronization may be used as a parameter for a Petri net entity composition. Instead of one operation of entity composition with access points α and β we will get several ones: ${}_\alpha\|{}_\beta^{\mathsf{or}}$, ${}_\alpha\|{}_\beta^{\mathsf{and}}$, ...,

*Here we use the obvious equation $\mathsf{and}(a_1, \mathsf{and}(a_2, a_3)) = \mathsf{and}(\mathsf{and}(a_1, a_2), a_3) = \mathsf{and}(a_1, a_2, a_3)$.

$\alpha \parallel_{\beta}^{\text{master}_i}$, etc .... In this case such composition rules can be easily represented as it is shown on Fig.10(b).

5   PN³-Tool.

The compositional specification model , CoPN, described in this paper is supported by a special Petri net tool called PN³-Tool. The detailed description of the tool can be found in (Anisimov it et al., 1994). In this section we will present the main features of PN³-Tool which can beapplied to multimedia specifications.

PN³-Tool has been developed on PC/MS Windows 3.x. Structurally, PN³-Tool consists of three levels: basic, algebraic and architectural editors.

The basic editor allows explicit creation of specifications in Petri net form. The editor includes standard Petri net editing facilities such as adding, removing, resizing Petri net elements (place, transitions, arcs, marking, labels). The possibility of referencing from either transition or place to another Petri net window, called *page,* allows for the specification of hierarchical structures. The example of basic Petri net editor is presented in Fig.11.

The second level is the algebraic editor, which presents the set of algebraic operations for Petri net specification. In the editor, pages are considered as operands for operations. The first and the most atomic operation is adding an atomic net which consists of a transition with an incoming and an outcoming place. These places are called *head* and *tail*, respectively, see (Kotov, 1978). On the basis of place merging, the following operations were implemented: sequential composition, choice, iteration, and disabling. The parallel composition can be performed either as a pure synchronization or as a transition merging operation through access points. So, the algebraic level makes it easy to construct a complex Petri net specification with from entities of regular structure.

At the top level is the architectural editor, which allows the user to manipulate Petri net entities in diagram form, i.e. in the form of interconnected boxes. Each Petri net entity must either refer to another entity configuration (another page) or directly to a Petri net. The editor of this level enables the designer to think in terms of entities which are represented by boxes. To compose two entities one only needs to connect corresponding access points. The user can produce specifications by either top-down or bottom-up approaches.T he example of architectural editor is depicted on Fig.12.

Useful additional features have also been implemented, which include the visual simulation of Petri nets, which can used for better understanding of the dynamic behavior and for analysis; and verification for several important safety properties (e.g. boundedness and freedom from deadlocks, etc.) by means of the Petri net cover tree building.
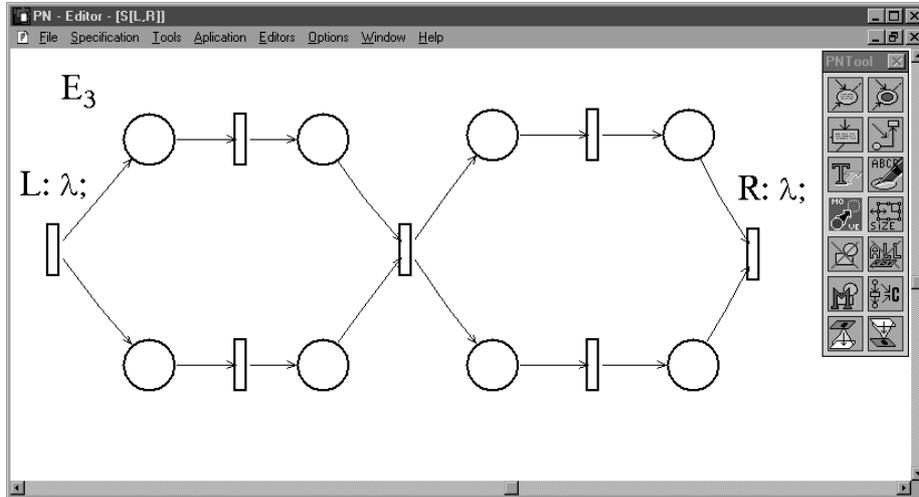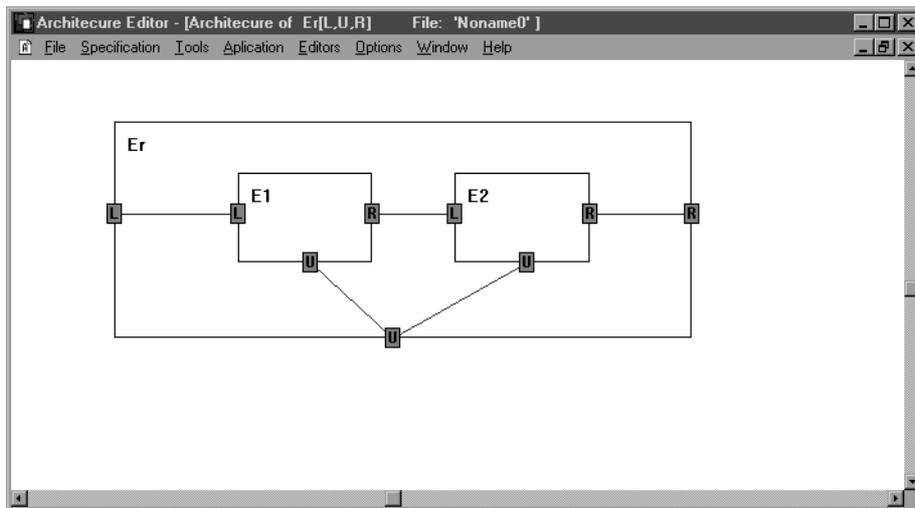
Fig. 11   Basic Petri net editor

Fig. 12   Architectural editor

6   Example: Specification of multimedia scenario.

To illustrate the practical applicability of the CoPN model in the specification of complex multimedia scenarios, we have developed a CoPN specification of the multimedia scenario example which was presented in (Vuong *et al.*, 1995). This example is interesting since while not overly complex, it involves several important multimedia synchronization requirements, including user interaction functions. In

order to simplify the specification we have mixed both Petri nets and diagram forms of representation. Fig.13 shows the top-level specification, which consists two independent parts: multimedia streams (mms) and user interaction modules (uim). While a designer may develop each part separately, at the end he must synchronize them together. First, we specify the internal representation of mms, and then deal with uim.

Multimedia streams. The scenario consists of two main streams, as shown in Figure 14(a): dynamic video and audio data, specified as entities VA_i; static data, specified as entities I_i and T_i. The static data includes two images I_1, I_2 and two texts T_1, T_2, shown in Fig.14(d). The pairs, image I_i with text T_i, must be "intra-synchronized" by means of head l and tail r access points.

The part of the scenario with dynamic data is represented by two sequential video clips (video streams) and one audio stream, as shown in Figure 14(b). The first video stream consists of a set of video frames $\{video_1, \ldots, video_n\}$, and the second stream, $\{video_{n+1}, \ldots, video_{2n}\}$, as shown in Figure 14(c). The audio stream is formed by the set of $\{audio_1, \ldots, audio_{2n}\}$. The specification resulting from intra-stream synchronization between VS_1 and VS_2 must in turn be inter-synchronized with the audio stream AS through the d and u access points, respectively. In inter-stream synchronization, transitions with the same labels (in this case, numbers from 0 to $2n + 1$) will be merged. Each transition in the MM entity has an additional access point $m:p$; necessary for stop/resume primitives.
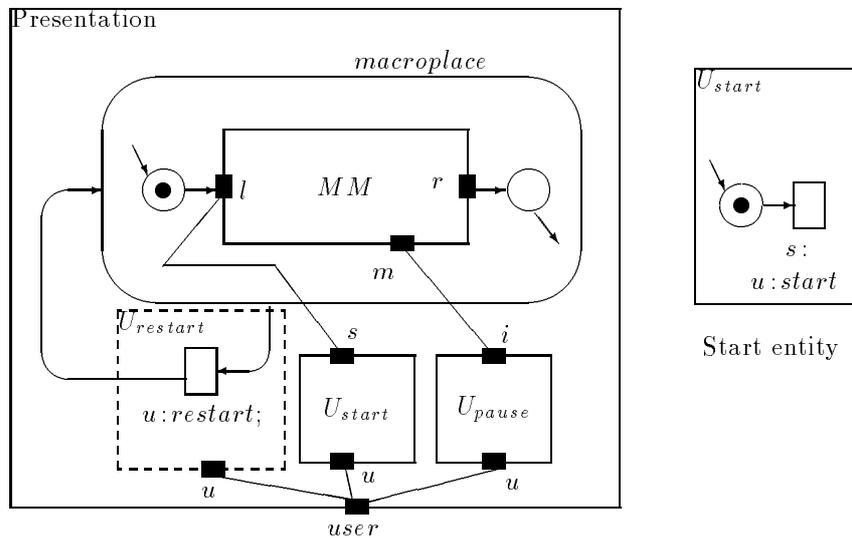

Fig. 13   Example of multimedia scenario specification with user interaction.

User interaction. As shown in Fig.13, the entity $Presentation$, is controlled

via the following types of user interactions at access point *user*:

- **start**: to perform an initialization and initiate the start of the multimedia streams. Entity U_{start} consists of one marked place and one transition which have to synchronize with the first transition of **mms**.
- **restart**: to allow the user to stop current multimedia presentation and start it from the very beginning. The *macroplace* can disrupt the activities of the MM entity at any time and restart the playing by means of the $u\!:\!restart$ transition.
- **stop** and **resume**: to specify the possibility to make a pause at any time and resume from where it was left of. The internal structure of the entity U_{pause} is specified in the left bottom corner of Figure 7.

The above example demonstrates the methodology for multimedia scenario specification by means of Petri net entities and macronets. This CoPN model allows the independent creation of separate parts of the specification and then their algamation into an overall specification in a compositional and hierarchical manner.

7   Conclusion.

In this paper, we propose a compositional and hierarchical approach to multimedia scenario specification using a compositional Petri net model, so called CoPN. This compositional approach enables the compact and readable specification of complex, large-scale specifications while preserving the fine granularity as well as supporting user interactions. While discussing the features of CoPN, we have focused on the compositional aspects of the multimedia synchronization specification, without explicitly addressing the time parameters of the specification. Preliminary ideas on incorporating the time stream Petri net (TSPN) in the compositional method for handling time are discussed. We are currently examining further the interplaying of times and compositions in Petri nets, and expect to report the complete results in a future report.

The PN³-Tool can also be enhanced in order to successfully support the specification of multimedia scenarios. The most significant modifications would include explicit implementation of macroplace operation "unwrapping"; flexible screen managing functions such as scaling and picture rotation; as well time constraint handling accordingly to the time model used.

8   Acknowledgments

(a) The main multimedia entity
and component entities.

(b) Internal representation
of VA_i streams.

(c) VS_i (video) and AS (audio) streams.

(d) I_i (image)
and T_i (text) streams.
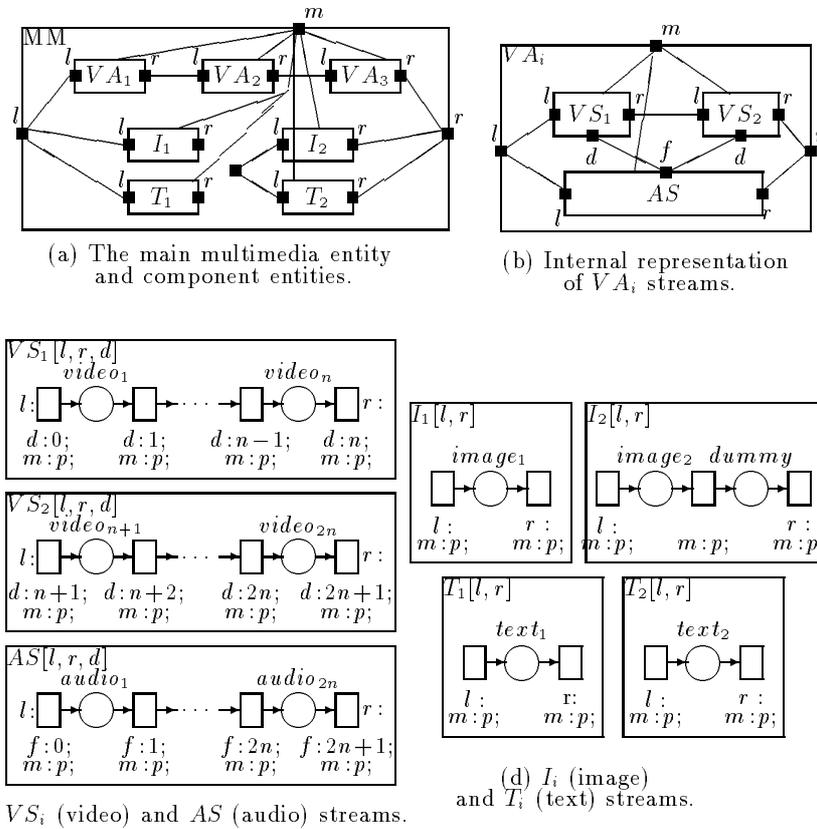
Fig. 14   Specification of multimedia streams.

Bibliography

Anisimov, N. A. (1989) "A Notion of Petri Net Entity for Communication Protocol Design", *Institute for Automation and Control Processes.*

Anisimov, N. A. (1991a) "A Petri Net Entity as a Formal Model for LOTOS, a Specification Language for Distributed and Concurrent Systems", in *Parallel Computing Technologies,* ed. N. N. Mirenkov (World Scientific), 440–450.

Anisimov, N. A. (1991b) "An Algebra of Regular Macronets for Formal Specification of Communication Protocols", *Computers and Artificial Intelligence,* **10**, 541–560.

Anisimov, N. A., (1991c) "A Notion of Entity Based on Petri Nets", in *Proceedings of the 1991 IFAC Workshop on Discrete Event System Theory and Applications in Manufacturing and Social Phenomena,* ed. Yu-Chi Ho, Ying-Ping Zheng, Shenyang, P. R. China, (International Academic Publisher) 142–147.

Anisimov, N.A., Kovalenko, A.A., Postupalski, P.A. (1994) "Compositional Petri Net Environment", in *Proc. of the 1994 IEEE Symposium on Emerging Technologies & Factory Automation: ETFA '94,* 420–427.

Anisimov, N.A., Kovalenko, A.A. (1995) "Towards Petri Net Calculi based on Synchronization via Places", in *Proc. of the 1995 IEEE Symposium on Parallel Algorithms/Architecture Synthesis* (Aizu, Japan) 264–270.

Anisimov, N.A., Koutny, M., (1996) "On Compositionality and Petri Nets in Protocol Engineering", in *Protocol Specification, Testing and Verification, XV.,* eds. P.Dembiński, M.Średniawa. (Chapman & Hall, 1996) 71–86.

Anisimov, N.A., Kovalenko, A.A., Postupalsky, P.A., Vuong, S.T., (1997) "Application of Compositional Petri Nets in the Specification of Multimedia Objects", in *Proceedings of the Fourth Pacific Workshop on Distributed Multimedia Systems (DMS'97),* Vancouver, Canada, (Knowledge Systems Institute) 52–59.

Best, E., Devillers, R., Hall, J.G. (1992) "The Box Calculus: a New Causal Algebra with Multi–label Communication", in *Advances in Petri Nets 1992,* ed. G.Rozenberg, *Lecture Notes in Computer Science,* **609** (Springer–Verlag) 21–69.

Brauer, W., Reisig, W., Rozenberg, G., (eds) (1987) "Petri Nets. Part I and II. Proc. of an Advanced Course, Bad Honnef", *Lecture Notes in Computer Science,* **254/255** (Springer–Verlag).

Cooper, K., (1995) "TSPN$_{ui}$: A Petri Net Model for Specifying User Interaction in Multimedia Presentations", *MASc Thesis, The University of British Columbia,* Canada.

Courtiat, J.-P., De Oliveira, R.C., Andriantsiferana, L., "Formal Modeling and verification of Multimedia Documents".

Diaz, M., Senac, P., (1993) "Time Stream Petri Nets: a model for multimedia streams synchronization", in *Proceedings of the International Conference on Multi-Media Modelling,* Singapore.

Kotov, V.E. (1978) "An Algebra for Parallelism Based on Petri Nets", *Lecture Notes in Computer Science,* **64** (Springer–Verlag) 39–55.

Little, T., Ghafoor, A., (1990) "Synchronization and storage models for multimedia objects", *IEEE Journal on Selected Areas in Communications,*Volume **8**, No **3**, 52–61.

Prabhakaran, B., Raghavan, S., (1993) "Synchronization Models Multimedia Presentations With User Interaction", *ACM Multimedia '93*, CA, 157–166.

Reisig, W. (1985) "Petri Nets: An Introduction", *EATCS Monograph on Theoretical Computer Science,* (Springer–Verlag).

Sénac, P., Diaz, M., Saqui-Sannes, P. (1994) "Toward a formal specification of multimedia synchronization scenarios", *Annals of Telecommunications,* Volume **49**, No **5-6** 297–314.

Sénac, P., Diaz, M., Léger, P., Saqui-Sannes, P., (1996) "Modeling Logical and Temporal Synchronization in Hypermedia Systems", *IEEE Journal of Selected Areas in Communications*, Volume **14**, No **1**, 84–103.

Vuong, S., Cooper, K., Ito, M. (1995) "Formal Methods for Modeling Multimedia Synchronization Requirements", in *Proc. of ICNP'95 - Int. Conf on Network Protocols*, (Tokyo, Japan).

Woo, M., Qazi, N., Ghafoor, A. (1994) "A synchronization framework forcommunication of preorchestrated multimedia information, multimedia objects", *IEEE Network,*.