Pergamon

Comput. & Graphics, Vol. 22, No. 6, pp. 735–743, 1998

PII: S0097-8493(98)00094-6
Printed in Great Britain
0097-8493/98/$ - see front matter

Technical Section

ANIMATION MODELING WITH PETRI NETS

LÉO P. MAGALHÃES†, ALBERTO B. RAPOSO and IVAN L. M. RICARTE

State University of Campinas (UNICAMP), School of Electrical and Computer Engineering (FEEC), Department of Computer Engineering and Industrial Automation (DCA), CP 6101-13083-970, Campinas SP, Brazil

Abstract—This paper introduces the use of Petri Nets as a modeling and analysis tool for animation environments. Firstly, the original formulation for Petri Nets is applied in two animation situations, one modeled as a state machine and another exploring interdependent transitions. Increasing the complexity level, some modeling extensions are discussed and more sophisticated animation examples are studied.

Key words: animation planning, computer modeled animation, Petri Nets, behavioral animation, animation modeling and analysis.

1. INTRODUCTION

Control modeling is one of the most important aspects in computer animation. It specifies how characters acting in an animation should move and how they interact with the animation environment. Movements of characters in an animation can be defined using parametrical interpolations, kinematic and dynamic equations (direct and inverse), genetic models, etc. [1, 3, 18, 20, 22]. Characters interacting with the environment can be controlled from an antecipative point of view (e.g., interpolation) or detected on the fly using for example collision detection techniques [6, 21].

At a more abstract level one can consider characters subject to emotions, having a reactive behavior, or being oriented towards a specific task or goal. In these cases, control can be based on strategies such as logical description of behaviors [8, 16], or systems of kinematic or dynamic equations [7].

This paper focuses on the support of animation modeling at this more abstract level using a formal framework based on Petri Nets theory. Besides being useful as a tool for modeling animation environments, Petri Nets also offer a powerful contribution for animation analysis.

This paper is structured as follows: Section 2 will introduce Petri Nets. Section 3 will present some initial examples of their use. Section 4 will address the use of Petri Nets in more sophisticated animation problems. The last sections will present the conclusions, next issues and related bibliography.

2. PETRI NETS: FUNDAMENTALS

Petri Nets [17] (from here on, PN) are a modeling tool applicable to a variety of fields and systems, specially suited to systems with concurrent events. Murata [15] presents a very good introduction to the theme. Formally, a PN can be defined as a 5-tuple (P, T, F, w, M_0), where: $P = \{P1, \ldots, P_m\}$ is a finite set of places; $T = \{t_1, \ldots, t_n\}$ is a finite set of transitions; $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; w: $F \rightarrow \{1,2,\ldots\}$ is a weight function; M_0: $P \rightarrow \{0,1,2,\ldots\}$ is the initial marking; with $(P \cap T) = \varnothing$ and $(P \cup T) \neq \varnothing$.

In a PN model, states are associated to places and marks, and events to transitions. A transition t is said to be enabled if each input place P_i of t is marked with at least $w(P_i, t)$ tokens, where $w(P_i, t)$ is the weight of the arc between P_i and t. Once enabled, a transition will fire when its associated event occurs. Firing the transition t removes $w(P_i, t)$ tokens from each input place P_i of t, and adds $w(t, P_o)$ tokens to each output place P_o of t.

A very useful notation for PN is the graphic notation (Fig. 1) which will be used in the examples throughout this paper. In this notation, circles represent places, bars represent transitions, dots the marks (also called tokens), and arrows the arcs, with weights above. By definition, an unlabeled arc has weight 1.

For example, in the PN of Fig. 1, only transition t_2 is enabled; t_1 is not enabled because it would require two marks in P_1 to fire, since $w(P_1, t_1) = 2$. When t_2 is fired, the marks in P_2 and P_3 are removed and P_4 receives one mark. It should be observed that the number of marks in a PN is not necessarily conserved.

Besides the graphical notation used in this paper, a matrix is also adequate to indicate the possible changes of states in a net. This matrix, C, has dimension $m \times n$, with position $C_{i,j}$ indicating how many tokens place P_i will receive (positive value) or lose (negative value) when transition t_j is fired. Representing a state by a vector marking

†Corresponding author. Tel.: +55-19-788-3706; Fax: +55-19-289-1395; E-mail: leopini@dca.fee.unicamp.br.

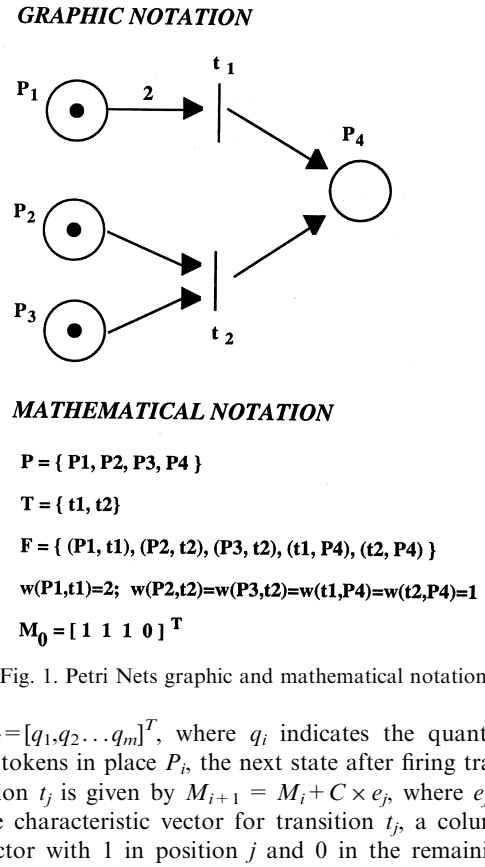

Fig. 1. Petri Nets graphic and mathematical notation

$M_i=[q_1,q_2 \ldots q_m]^T$, where q_i indicates the quantity of tokens in place P_i, the next state after firing transition t_j is given by $M_{i+1} = M_i + C \times e_j$, where e_j is the characteristic vector for transition t_j, a column vector with 1 in position j and 0 in the remaining positions.

Summarizing, the behavior of a system using PN is described in terms of its states and their changes [15]. States are modeled by *places* and *marks*, which define the current state of the system. *Transitions* (firing rules) model the dynamic behavior of the system. *Arcs* indicate the sequence of possible transitions between states and they can be *weighted* meaning the quantity of *marks* necessary to fire a transition.

Besides the modeling capabilities of PN, their support for analysis is very important and useful. This analysis is based on the properties of the mathematical model of PN. Some of these properties are:

Reachability: is there a sequence of firing that reaches a given state?

Boundness: will a place be overloaded? A PN can be defined as k-bounded if the number of tokens in each place does not exceed k.

Liveness: is there any state or sequence of states which will not be reached anymore, indicating a possible deadlock?

Reversibility: is it possible to return to a defined initial state M_0?

Persistence: is the firing of any pair of enabled transitions interdependent, i.e., the firing of one will disable the other?

Synchronic Distance: what is the relationship between two transitions? This metric is related to the degree of mutual dependence between transitions.

3. MODELS FOR SIMPLE ANIMATIONS

The correct modeling and use of PN properties for analysis allow an animator to preview the behavior of an animation even before starting any shot. *Reachability* can be used to detect modeling problems related to defined animation states which will never be reached. *Boundness* is related, e.g., to a number of actors wished in a state. *Liveness* can be used to find states which will never happen if an specified state is reached. *Reversibility* allows to test whether an initial state can be reached from another state. *Persistence* and *synchronic* distance test the interdependence between animation events.

The following two sections introduce the powerful characteristics of PN applied to animation problems by means of simple but clear examples. The benefits that can be achieved in more complex environments will be discussed in Section 4.

3.1. *Single sphere example*

The example of Fig. 2 shows two buttons and a sphere which can follow two different trajectories depending on which button was chosen. One button is associated to an internal trajectory of the sphere and the other with an external one.

For the above example the event of pressing one button could be associated to a warning signal (e.g., button 1 danger, button 2 no problems), signalizing the trajectory to be followed.

The animation is modeled in PN as follows:

Place 1 (P_1) is associated to trajectory A1
Place 2 (P_2) is associated to trajectory A2
Transition 1 (t_1) is associated to the press of button 1
Transition 2 (t_2) is associated to the press of button 2.

Formally, the PN for this example is a 5-tuple (P, T, F, w, M_0), where: $P = \{P_1, P_2\}$ is the set of places; $T = \{t_1, t_2\}$ is the set of transitions; $F = \{(P_1, t_2), (P_2, t_1), (t_1, P_1), (t_2, P_2)\}$ is the set of arcs; $w(f) = 1$ **for all** $f \in F$; $M_0=[1\ 0]^T$ is the initial marking.

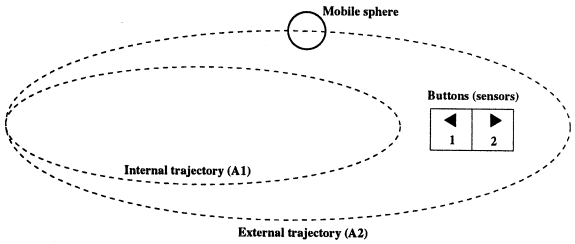

Fig. 2. Basic example

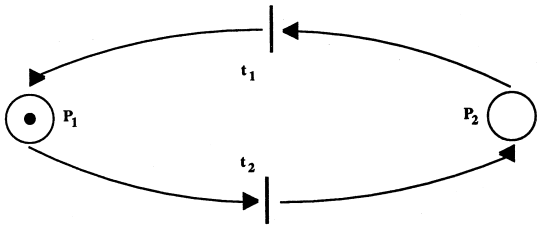

Fig. 3. PN representation of the basic example

Figure 3 shows the graphical representation of the PN model for this animation. The main character of the animation, the sphere, is represented by the mark (dot token) in the Fig. 3.

The graph gives the reader the following direct information. First, there are two stationary states in the environment. When the mark is in place P_1, the sphere follows trajectory A1. Otherwise, when the mark is in place P_2, the sphere follows trajectory A2. Second, the effect of pressing a button is dependent on the current trajectory. For example, if the sphere is following A1 (token in P_1) the press of button 1 has no effect, since t_1 is not enabled. Finally, the movement will remain forever, since the graph does not present any state where the sphere is not moving.

Information can also be obtained by means of an informal analysis of the graph based on the properties of PN:

- All possible states can be reached. The initial state $M_0=[1\ 0]^T$ after the firing of t_2 will be transformed into $M_1=[0\ 1]^T$, which after the firing of t_1 will return to M_0, and so on.
- There are no deadlocks, i.e., the graph is alive. All the possible states (M_0 and M_1) can be reached.
- Whichever the initial state is, it is possible to return to it.
- The transitions are mutually dependent, i.e., it is not possible to fire one of them twice without firing the other. Even though one can press the same button two times sequentially, the second time has no effect in the animation.

This very simple example illustrates the basic mapping between situations in an animation and a corresponding PN model. In the following section the complexity of the presented example will be increased in order to stress the use of PN for the analysis of an animation behavior.

3.2. *Two spheres example*

This section will detail some additional modeling and analysis aspects of PN taking a more complex environment.

The basic example, Fig. 2, will be modified as shown in Fig. 4. Now there are two spheres, each one travelling in one of the two possible trajectories (internal or external) controlled by a button. The animation has a behavioral restriction defined by the rule that only one of the spheres can be at its external trajectory at a time in order to avoid collisions.

Figure 5 introduces the PN model for this example. The following characteristics are modeled:

- There are four places defining the two possible trajectories for each sphere, $P = \{A_1, A_2, B_1, B_2\}$.
- There are four transitions defining button press events, $T = \{t_1, t_2, t_3, t_4\}$,where t_1 and t_4 are associated to the press of the buttons that put the spheres A and B, respectively, in their internal trajectories, and transitions t_2 and t_3 are associated to the press of the buttons that put the spheres A and B, respectively, in their external trajectories.
- All arcs have unit weight.

Let the initial marking be $M_0=[1\ 0\ 1\ 0]^T$. It can easily be seen that in this state only transitions t_2 and t_3 are enabled, and that firing t_2 will lead to a state $M_1=[0\ 1\ 1\ 0]^T$ whereas firing t_3 will lead to $M_2=[1\ 0\ 0\ 1]^T$. From M_1, the only possible transition, t_1, takes the net back to M_0. The same happens from M_2, for which the only possible transition is t_4, which also takes the net back to the initial marking when fired. Therefore, the complete set of states for this net is $M = \{M_0, M_1, M_2\}$.

A powerful tool for the analysis of PN is the coverability graph which offers a vision of the complete sequence of transitions and states in a PN. Figure 6 illustrates the coverability graph for this example, considering the initial state M_0.

Based on the coverability graph and taking into account the PN properties, the following can be stated:

Reachability: Starting at one of the states of M the system never goes to the forbidden state $[0\ 1\ 0\ 1]^T$, which might cause a collision, or to the impossible states $[1\ 1\ 0\ 0]^T$ and $[0\ 0\ 1\ 1]^T$, in which one sphere would follow two or no trajectories.

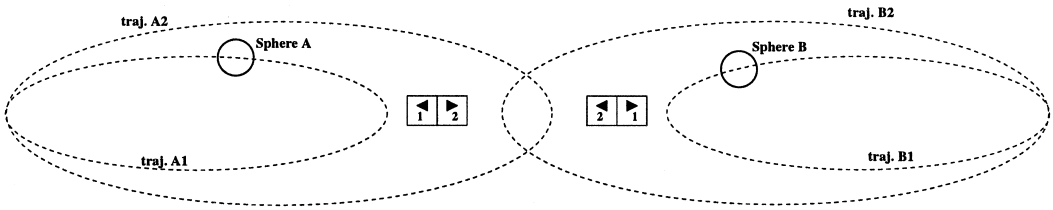

Fig. 4. Two spheres example

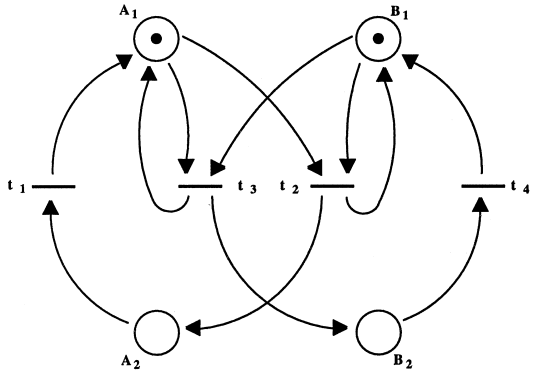

Fig. 5. PN for the example with 2 spheres

All other states can be obtained by a given firing sequence.

Boundness: the PN is 1-bounded, i.e., no place will have more than one mark at a time. A PN with this property is called *safe*.

Liveness: each valid state has a following valid state for a firing sequence, i.e., there are no deadlocks. In addition, all transitions appear in the coverability graph, meaning there are no *dead* transitions.

Reversibility: it is always possible to return to an initial state by a firing sequence.

Persistence: This PN is not persistent. The firing of t_2 will inhibit t_3 and vice versa. This expresses the mutual exclusion relationship between both events.

Synchronic distance: this characteristic expresses the level of mutual dependence between two transitions. Considering σ a firing sequence at any marking in M and $\sigma(t_i)$ the number of times t_i fires in σ, $d_{i,j} = \max|\sigma(t_i) - \sigma(t_j)|$.

Regarding the synchronic distance, $d_{1,2} = d_{3,4} = 1$ shows that these pairs of transitions are interdependent—in fact, the transitions of each of these pairs are associated to events of the same sphere. On the other hand, $d_{1,4} = d_{2,3} = \infty$ shows that these pairs of transitions are associated to independent events, which should be true since they are associated to events of different spheres.

It can be concluded from the above points that:

1. The developed model has no undesired states. It conforms with the initial animation description.

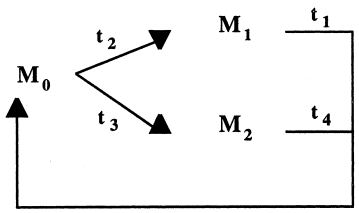

Fig. 6. Coverability graph for the example

2. There are no deadlocks, since the control statements will always put the animation in a valid state.
3. The animation will run forever, since there is no final state.
4. There are two mutually excludent transitions, confirming the behavioral restriction.
5. The only mutually dependent transitions, with $d_{i,j} = 1$, are associated to events of the same object, either sphere 1 or sphere 2.

The PN model for these animations associates transitions with user intervention (pressing buttons). If autonomous behavior were required, PN extensions would have to be used, as described in the next section.

4. PN EXTENSIONS FOR COMPLEX ANIMATION ENVIRONMENTS

The previous section introduced the use of PN in animation environments. The first example presented a state-machine PN which is a subclass of PN for which each transition has exactly one input and one output. The second example introduced interdependent transitions in a more complex environment.

In this section, the reuse of PN models will be emphasized by means of an example where the simple model of Section 3.1 is reused in a typical computer animation situation. After that, the notion of time is introduced in PN by the use of a PN extension.

4.1. *Dancer example*

A classical example in the computer animation field is the control of the walking movement of a biped structure such as a human being. The walking movement must satisfy the constraint that both legs cannot be out of the ground at the same time. This movement can be modeled by the PN of Fig. 5 considering that each ball now represents a leg, the internal trajectory represents the leg on the ground, and the external trajectory represents the leg off the ground. However, the walking movement also requires the legs to be raised alternately. (It does not make sense to raise the same leg two times sequentially.) Due to this additional restriction, the walking movement of a biped can be now easily modeled by a state machine similar to that of Fig. 3, with four states: left leg up, left leg down, right leg up, and right leg down.

A more challenging example is to define the movements of a ballet dancer. In this model, the dance consists of two basic and nonsequential movements: walking and jumping. In addition, the dancer can tiptoe or have the feet in the normal position. The supposed choreography requires that the dancer jumps only when tiptoeing.

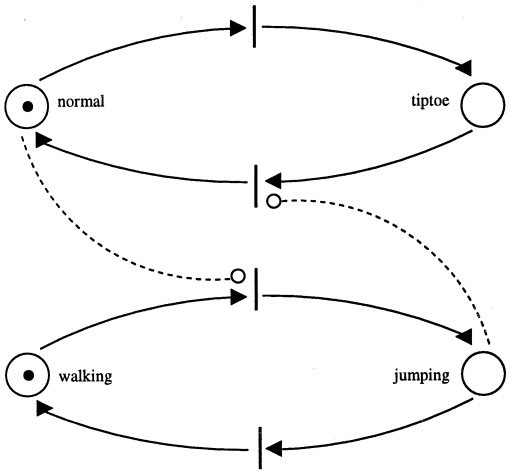

Fig. 7. PN graph for the dancer example

The above situation can be modeled using two PN similar to that of Fig. 3, one modeling the feet position and another modeling the dancer's movement. The PN are interconnected by inhibitor arcs, a tool to synchronize interdependent events [10, 13]. In Fig. 7, inhibitor arcs are represented by dashed lines with a circle at the end. This kind of arc enables the inhibited transition only when its input place becomes empty. The inhibitor arcs connecting the PN guarantee that the dancer will jump only when tiptoeing.

This example illustrates not only the reuse of a basic PN model, but also the capacity of encapsulation offered by PN. The graph of Fig. 7 hides the details of the walking and jumping movements, enforcing a hierarchical description model. The movement of walking, for example, is modeled by a PN state machine as commented before. A further step in this hierarchical description could be the definition of the relationship among the various dancers in a ballet performance, hiding the control of each dancer, as discussed in the next section.

The models presented up to this point do not use the notion of time, e.g., it is not possible to control the duration of the walking or jumping movements. The notion of time is introduced by a PN extension presented in the next section.

4.2. *On stage example*

Generalized Stochastic Petri Nets are derived from PN by associating fire rates to transitions and partitioning the set of transitions into two subsets [13]. This technique enlarges the class of animations that can be modeled by PN introducing the notion of time associated to a transition fire, as will be seen next.

4.2.1. *Generalized stochastic PN*. First of all the PN definition of Section 2 is extended by introducing the set of firing rates, possibly marking-dependent, associated with the PN transitions. For a PN with s transitions the set $R = \{r_1,\ldots,r_s\}$ is defined, where r_i is the rate associated with the firing of transition t_i. This formulation defines the Stochastic Petri Nets (SPN) [2, 5, 14].

Generalized Stochastic Petri Nets (GSPN) [13] are obtained by generalizing SPN, subdividing the set T (see Section 2) into timed and immediate transitions, reducing R to s' elements, where s' is the number of timed transitions. Immediate transitions fire in zero time after enabled, while timed transitions fire after a random exponentially distributed enabling time, introducing an additional and powerful tool for modeling animation environments.

The following statements apply to GSPN:

- If the set of simultaneously enabled transitions H comprises only timed transitions, the enabled timed transition t_i fires with probability $r_i/\Sigma_{k\epsilon H} r_k$.
- If the set of simultaneously enabled transitions comprises both kind of transitions only the immediate ones can fire. In that case if there are more than one immediate transition enabled, it is necessary to associate a probability function to define which one will fire.

In this paper double bars represent timed transitions and immediate transitions are represented by a single bar as in the basic PN graphical representation.

4.2.2. *Modeling*. This section explores some of the additional modeling features offered by GSPN to animation environments.

The following example presents a dance performance with several dancers, each one modeled as described in Section 4.1, sharing a limited number of costumes. Dancers enter and leave the stage at random rates. Two dancers cannot share a costume at the same time and, if all costumes are in use, a dancer has to wait until one of the dancers currently performing finishes in order to get a costume.

Figure 8 is the PN graph modeling the described behavior. The model presents the following states and transitions:

P_1 dancers that are going to perform

P_2 dancers ready to perform

P_3 costumes that are currently available

P_4 dancers on stage, each one with a different costume

P_5 dancers ready to perform but waiting for a costume

t_1 enables dancer

t_2 dancer starts performing

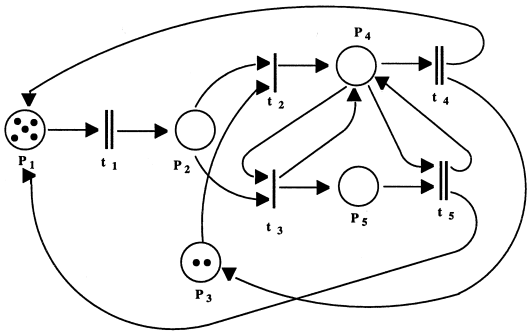

Fig. 8. PN graph for the dancers and costumes example

t_3 dancer has to wait for a costume

t_4 dancer has finished performing

t_5 dancer has finished and is substituted by another one that was waiting for his/her costume.

It is important to note that t_2 and t_3 are immediate transitions firing when their preconditions are fulfilled. On the other hand t_1, t_4, and t_5 are timed transitions with associated random exponentially distributed firing rates.

The initial state is $M_0 = [5\ 0\ 2\ 0\ 0]^T$ defining an environment composed by five dancers and two costumes.

Complementing the ideas discussed by the examples of Section 2, the current example will be analysed by means of a simulation, studying the effect of varying the firing rates in the animation behavior.

4.2.3. *Simulation analysis*. The presented example will be simulated to verify a condition to be avoided: $P_3 > 0$ and $P_5 > 0$. This condition represents what can be considered a bad system behavior, because there is at least one dancer waiting for performing ($P_5 > 0$), although there is (are) available costume(s) ($P_3 > 0$).

For this simulation the following parameters will be considered. λ_1 is the average value of an exponentially distributed random variable that determines the interval between firings of t_1, weighted by q_1 (the number of tokens in P_1). λ_4 and λ_5 are the average values of exponentially distributed random variables that control the interval between successive firings of t_4 and t_5, respectively. The value λ_i defines an average firing rate $1/\lambda_i$.

In the simulation, when both immediate transitions (t_2 and t_3) are enabled, it is defined that t_2 will fire.

In order to demonstrate one of the possible analyses to predict the system behavior, some simulation data will be graphically shown.

Figure 9 shows the behavior of the system for λ_1, λ_4, and λ_5 varying between 0.25 and 2.0 time units. The best system behavior of the current simulation is obtained for $\lambda_4 = 2.0$ and $\lambda_5 = 0.25$. This figure suggests that the rate λ_4/λ_5 defines part of the system behavior. However, the system is also sensitive to the variation of λ_1 as shown by the dashed curves.

This lead up to a further simulation presented in Fig. 10, now including the influence of the rate λ_1/λ_5. For this simulation, the values of λ_4/λ_5 and λ_1/λ_5 are between 0.125 and 8.0. This figure allows a more precise analysis of the system behavior. It can be seen that the system has a worst behavior peak in the region where λ_4/λ_5 and λ_1/λ_5 are smaller than one—for $\lambda_4/\lambda_5 = 0.125$ and $\lambda_1/\lambda_5 = 0.5$—and good behavior in the regions where at least one of these rates is high.

Looking closely the obtained simulation data for the peak region, the following relationships were obtained: $\lambda_4/\lambda_5 < 1$, $\lambda_1/\lambda_5 < 1$, and $\lambda_5 > \lambda_4 > \lambda_1/q_1$ (the firing rate of t_1 is weighted by q_1). This means that, in the region of interest, the rate which causes dancers to become ready to perform (q_1/λ_1) is higher than the one which causes dancers to leave the stage ($1/\lambda_4$). This contributes to put dancers in P_5 (waiting to start performing). Furthermore, the

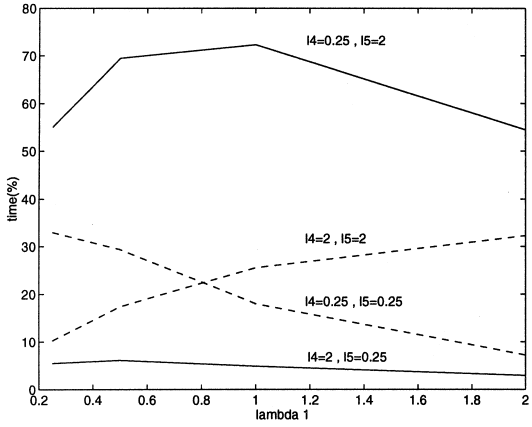

Fig. 9. Percentage of time where $P_3 > 0$ and $P_5 > 0$. In the figure *li* symbolizes λ_i

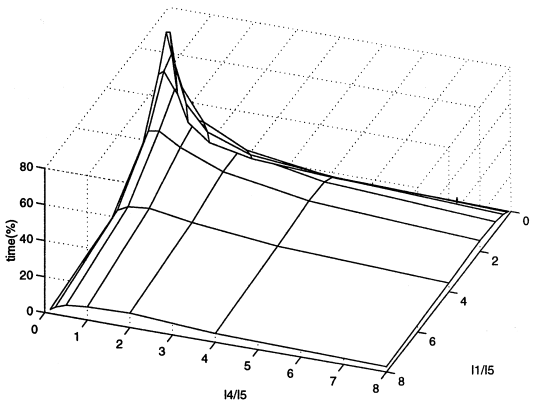

Fig. 10. 3D view of simulation results

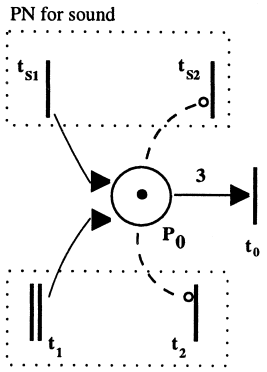

Fig. 11. Sound synchronization example

firing rate of t_4 is higher than that of t_5. This causes dancers in P_5 to remain there although costumes became available, because t_5 substitutes a dancer for another in the performance, while t_4 simply removes dancers.

This example illustrates the use of PN to model situations involving concurrency, synchronization and conflict resolution. These characteristics are also present in many other situations commonly analysed through PN, such as in multiprocessor systems [13].

4.3. *Other PN extensions*

In order to show other modeling tools, the example of the previous section will be enhanced by establishing that the beginning of the dance has to simultaneously start a sound track. This can be done by using inhibitors arcs, as shown in Fig. 11. In the example, after t_1 sends its first token (indicating that the first dancer is ready) and the sound PN signalizes sound is ready (i.e., t_{S1} fires), then P_0 has three tokens and t_0 fires. After that, both sound (t_{S2}) and movement (t_2) are allowed to proceed.

A second example also derived from that of the previous section is defined by redirecting the arcs from t_4 and t_5 (previously going to P_1) to a new node and then synchronizing the beginning of the next dance with the arrival of the fifth token (or dancer). In this case, the next performance may begin only when all the dancers of the previous one have finished dancing. Figure 12 shows the representation of this sequencing mechanism.

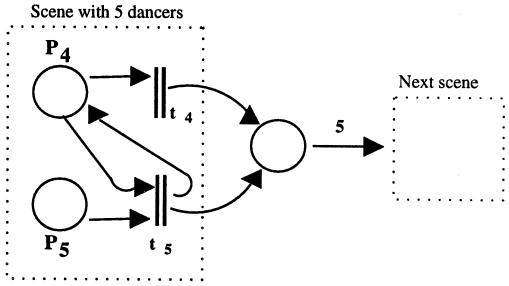

Fig. 12. Sequencing example

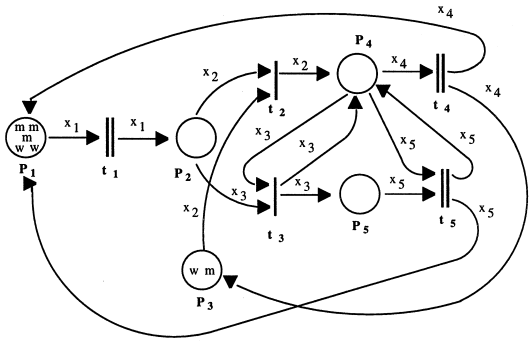

Fig. 13. Example using a Predicate/Transition Net

This second example may exhibit a deadlock situation if a last mark remains in P_5, because there will not be another mark in P_4 to enable t_5. A simulation similar to that of the previous section would be essential to avoid such situation.

A third example using once more the example of the previous section presents the use of Predicate/Transition Nets [11] which allows the differentiation of tokens, defining a type for them. The arcs have labels defining variables that dictate how many and which kind of tokens will be removed from or added to the places. The same variable appearing in the incoming and outgoing arcs of a transition denotes the same token type. A transition is enabled if there is at least one possibility of consistent substitution of variables into typed tokens.

So, in the example (Fig. 13) it is possible to distinguish male from female dancers, respectively tokens labeled m and w. The same is valid for the costumes, that can be masculine (m) or feminine (w). It is important to note that transition t_2 is enabled only if there is an adequate costume for the waiting dancer, which is indicated by the same value for label x_2, m or w, in both incoming arcs. The same is valid for t_5, that can substitute a dancer only for another of the same sex. The different indexes used for arcs leaving or entering a transition express that the token type is defined each time a transition fires. For instance, after t_1 fires twice (m and w), the firing of t_2 can get from P_2 m or w depending exclusively on t_2.

Another way to include the notion of time in PN, besides SPN and GSPN, is by the use of timed PN [19]. This PN extension defines a holding time for tokens in a place before the enabling of its output transition(s). It can be considered a simplification of SPN where the firing rates are constants and not exponentially distributed random variables.

5. CONCLUSION

This paper introduced the use of PN and some of its extensions for animation modeling and analysis. Another of the few attempts to apply PN in Computer Graphics uses them in an interface environment [4].

As other modeling techniques, PN fits very well when modeling the different aspects of an animation behavior. The problems faced when modeling an animation environment are quite similar to those where PN techniques are broadly used, they present common characteristics such as concurrency, synchronization, and conflict resolution.

PN also offers some additional mechanisms not present in most computer animation modeling techniques which enhance the process of animation development:

- The graphic representation encapsulates non-desired details, offering a very clear hierarchical description model.
- PN offers a very strong theoretical support for process analysis. Section 2 has shown some of the possible analysis that can be performed based on a graph model. Additional techniques not discussed in this paper are also available, such as state equations and priority net [15].
- Simulation techniques as presented in Section 4 complement the system analysis.

Several other tools and extensions related to PN have been developed. Some of them are: Decomposition and Aggregation (to treat the explosion of states in a SPN and GSPN) [23], colored PN [12], and other strategies allowing firing time of transitions to be specified by an arbitrary time distribution function [9].

As seen by the presented examples, PN favours the reuse of animation models. For instance, the first example (Fig. 3) represents the class of all animations in which the animated object can assume one of two possible states—as in the case of the two-orbit sphere or the feet position of a walking biped. Furthermore, the use of hierarchical PN supports encapsulation, hiding animation control details of lower level models.

Such features of animation model by PN motivates the creation of libraries with primitive PN blocks (graphs) that may be used by the animators to build the scripts that control the behavior of the animation. By using these primitives, animation modelers would not have to build the PN model for each animation from scratch, but rather they would identify animation patterns and interconnect them to define the desired behavior, as done in the case of a single dancer (Fig. 7). Furthermore, the use of PN simulators could help the animator to analyse and probe the animation model.

More interestingly, PN offers the possibility to anticipate and test animation behavior even before a single frame is shot.

For all of that, the authors believe that the introduction of PN and its extensions can bring to the field of Computer Graphics a large and well established set of techniques for animation modeling and analysis.

Acknowledgements—Part of this research was developed during the stay of the first two authors in the University of Waterloo—Computer Graphics Laboratory, Computer Science Department. The authors would like to thank the following institutions for the expressive support granted to this research: UNICAMP (State University of Campinas), FAPESP (Foundation for Research Support of the State of São Paulo) and the University of Waterloo. Thanks also to J. T. F. de Camargo and J. L. E. Campos for their valuable comments and contributions.

REFERENCES

1. Badler, N. I., Computer Animation Techniques. In *Introduction to Computer Graphics, Course Notes for SIGGRAPH 95*, Vol. 21. ACM-SIGGRAPH, 1995.
2. Balbo, G., On the Success of Stochastic Petri Nets. In *Proc. 6th Int. Workshop on Petri Nets and Performance Models*. IEEE, 1995. ISBN 0-8186-7210-2, pp. 2–9.
3. Barzel, R., *Physically-Based Modeling for Computer Graphics: a Structured Approach*. Academic Press, Inc., 1992. ISBN 0-12-079880-8.
4. Bastide, R. and Palanque, P., A Petri-Net based environment for the design of event-driven interfaces. *Lecture Notes in Computer Science*, 1995, **935,** 66–83.
5. Bause, F. and Kritzinger, P.S., Stochastic Petri nets: an introduction to the theory. *Advanced Studies in Computer Science*. Verlag Vieweg, 1996. ISBN 3-528-05535-9.
6. Camargo, J. T. F., Magalhães, L. P. and Raposo, A. B., Modeling motion simulation with DEDS. In *Proceedings of IFIP 94—13th: Congress of the International Federation of Information Processing*, 1994. http: //www.dca.fee.unicamp.br/ projects/prosim/publiPS.html, pp. 162–167.
7. Camargo, J. T. F., Magalhães, L. P. and Raposo, A. B., Foundations of Computer Modeled Animation. In *SIBGRAPI 95—Brazilian Symposium of Computer Graphics and Image Processing*, 1995. http://www.dca.fee.unicamp.br/ projects/prosim/publiPS.html (in Portuguese).
8. Costa, M. and Feijó, B., Agents with emotions in behavioral animation. *Computer & Graphics*, 1996, **20**(3), 377–386.
9. Dugan, J.B., *Extended Stochastic Petri Nets: Applications and Analysis*. Ph.D. Thesis, Dept. of Electrical Engineering, Duke University, 1984.
10. Dugan, J. B., Trivedi, K. S., Geist, R. M. and Nicola, V.F., Extended Stochastic Petri Nets: Application and Analysis. In *Proc. PERFORMANCE '84*, Paris, France, 1984.
11. Genrich, H. J., Predicate/Transition Nets. In *High-level Petri-Nets: Theory and Application*. Springer-Verlag, 1991. ISBN 3-540-54125X, pp. 3–43.
12. Jensen, K., Coloured Petri Nets: a High Level Language for System Design and Analysis. In *High-level Petri-Nets: Theory and Application*. Springer-Verlag, 1991. ISBN 3-540-54125X, pp. 44–119.
13. Marsan, M. A., Conte, G. and Balbo, G., A class of generalized stochastic Petri Nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 1984, **2**(2), 93–122.
14. Molloy, M. K., Performance analysis using stochastic Petri Nets. *IEEE Transactions on Computers*, 1982, **31**(9), 913–917.
15. Murata, T., Petri Nets: properties, analysis and applications. *Proceedings of the IEEE*, 1989, **77**(4), 541–580.

16. Perlin, K. and Goldberg, A., Improv: a System for Scripting Interactive Actors in Virtual Worlds. In *Proceedings of SIGGRAPH '96*, 205–216, 1996.
17. Petri, C. A., *Kommunikation mit Automaten*. Schriften des IIM Nr. 3, Bonn: Institut fuer Instrumentelle Mathematik, 1962.
18. Preston, M. and Hewitt, W. T., Animation using NURBS. *Computer Graphics Forum*, 1994, **4**(13), 229–241.
19. Ramamoorthy, C. V. and Ho, G. S., Performance evaluation of asynchronous concurrent systems using Petri Nets. *IEEE Transactions on Software Engineering*, 1980, **6**(5), 440–449.
20. Sims, K., Evolving Virtual Criatures, In *Proceedings of SIGGRAPH '94*, 15–22, 1994.
21. Thalmann, N. M. and Thalmann, D., Complex models for animating synthetic actors. *IEEE Computer Graphics and Applications*, 1991, **11**(5), 32–44.
22. Wyvill, B., A Computer Animation Tutorial. In *Computer Graphics Techniques: Theory and Practice* (eds D.F. Rogers and R.A. Earnshaw), pp. 235–282. Springer-Verlag, 1990.
23. Ziegler, P. and Szczerbicka, H., A Structured Based Decomposition Approach for GSPN. In *Proc. 6th Int. Workshop on Petri Nets and Performance Models*, pp. 261–270. IEEE, 1995. ISBN 0-8186-7210-2.