University of Castilla-La Mancha

A publication of the

Department of Computer Science

Simulation and visualization of Timed-Arc Petri Nets in TPAL

by

Javier Ulibarri López, Valentín Valero Ruiz, Juan José Pardo Mateo and M. Emilia Cambronero Piqueras.

Technical Report **#DIAB-04-10-1** October 2004

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

1 Introduction

Nowadays the complexity of many computer systems is very great, and it could be greater if designed for critic tasks, that may be a loose of money, of reputation and even of lives, and is in these cases when necessity to ensure that the system is correct (Medical Systems) and fulfills its task efficiently (Realtime systems) arises with extreme importance. Thus, has been required the development of formal models that allow to study with rigor these systems. Within the formal models we could distinguish two great types or groups:

- Algebraic models, based on a structured language with a set of operators.
- Models based on "graphical representations", which allow us to represent both the structure and the behaviour of the modeled system.

It is clear that each group of models will have its advantages and their disadvantages. The advantage that algebraic models can provide us is that they use a similar language to programming languages for its definition, which facilitates the specification of the system to software designers. On the other hand, the main advantage of graphical models is that they facilitate their interpretation, in addition to the existence of more comfortable and less tedious methods to verify the system that using process algebras.

Once defined the formal framework, and with the goal to facilitate the specification, design and verification tasks, many tools have been developed, which work with some of the mentioned formalisms (UPPAAL [BLPY95], CADP, LOLA (Lotos Laboratory)[DIT95]).

It is interesting to be able to benefit from the advantages that these different models offer us, and it is in this framework where the TPAL tool is, automatically translating from a specification to some specific formalisms. From a formal definition that has been made in an algebraic language (TPAL), the tool of the same name makes a translation to dynamic state graphs [PVCC01], timed-arc Petri nets [VPC02] and timed automata, thus we just need to define the concurrent system once, and then we can evaluate the system properties by using the more appropiate formalism according to the properties of interest.

TPAL tool (*ftp://penelope.info-ab.uclm.es/TPAL,http://www.info-ab.uclm.es/fmc/tpaltool/tpaltool.htm*) is being developed by our research group *(http://www.info-ab.uclm.es/fmc)*; this tool supports the specification of timed and probabilistic concurrent systems, began to be developed in 1997, and simultaneously to its development the formal framework in which is based has been developed, such as the dynamic state graphs model, and the model of timed-arc Petri nets.

TPAL is available for Linux systems under platform IBM/PC, as well as Solaris under platform Sun Sparc.

Their operation requirements are scarce, only requiring as additional software the graphical user interface toolkit XForms.

In addition, it is advisable to have a great amount of RAM memory, since the size of the generated graphs can be high and they stay in main memory.

In the previous version of the tool, timed-arc Petri nets were automatically generated from the specifications written in the algebraic language TPAL, but only a textual visualization of them was provided (see Figure 1).

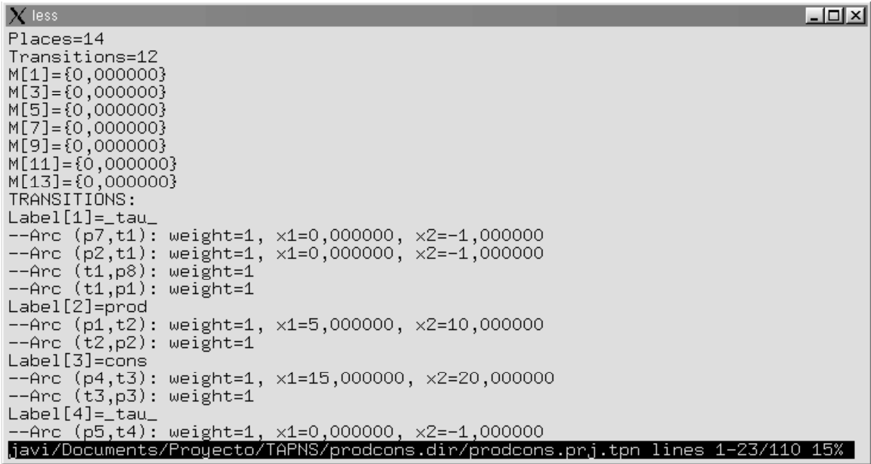

Figure 1: Textual visualization of a time-arc Petri net in TPAL

In Figure 1 we can observe how unfriendly it was to see a timed-arc Petri net, as one of the advantages that graphical models provide us in comparison to the algebraic ones, is the easy and intuitive it is to interpret them. The need to create a graphical viewer for these nets was evident.

Furthermore, it is also important to be able to simulate the behaviour of the generated nets, thus allowing the user to see how the system can evolve either by time elapsing or by the firing of transitions.

1.1 New features of TPAL

Basically, the external aspect of the present version of TPAL has not significant modifications with respect to the previous version. The efforts have been centered in extending the functionality of the tool, concluding the graphical viewer and simulator of timed-arc Petri nets.

The visualization and simulation are integrated through an only interface.

Thus, we have the following new functions in TPAL:

- Visualization in a graphical way the timed-arc Petri nets obtained by the translation of the TPAL specifications.
- The user can move the different net elements (places or transitions) using the mouse.
- The user can obtain detailed information of any net element, just clicking on it with the mouse.
- The user can make a simulation of the net, which can be automatic, semi-automatic or manual.
- We can obtain a trace of the simulation.

Furthermore, the user can modify many parameters that affect the visualization and the simulation, more specifically we can configure:

- The size and colour of the places.
- The size and colour of the transitions. It is also possible to define the colour associated to each possible transition state (enabled or not enabled).
- The width and colour of the arcs. It is possible to configure place-transition arcs independently form transition-place arcs, as they are different in this model.
- If arc weights are shown or not in the visualization.
- If time-intervals are pictured or not in the corresponding place-transition arcs.
- Configuration of the canvas where the net is painted.

Concerning with the simulation, we have the following parameters:

- The seed for the pseudo-randomization.
- Avoidance of time-locks, i.e., the system must evolve trying to avoid that some tokens become too old to be used to fire any transition.
- The maximum simulation time.
- The maximum number of steps for the simulation.
- The simulation speed.

This enumeration of functions will be seen in detail later, in a more practical and visual way, when the particularities of the net representation are explained.

1.2 Example

In this section we illustrate these new functionalities with the following example. It is a typical producer/consumer system, in which we will have a *buffer* of capacity 5.

The project file for this example is shown in Figure 2, and the cooresponding process specification file is shown in Figure 3.

```
Processes  ProdCons;

-- Project specification, the name must coincide with that
-- of this file, without the extension:
Project prodcons;
   -- List of processes or components:
   Processes ProdCons;
   -- List of synchronized actions
   Sync ;
   -- List of hidden actions
   Hidden ;
End_project
```

Figure 2: Project file for the P/C system

```
-- Producer/Consumer System
SPECIFICATION ProdCons [ prod, cons, ins, extr ];
BEHAVIOUR
     ((Producer [] Consumer) [ins, extr] Buffer) \ins \extr
WHERE
     PROCESS Producer [ prod, ins ];
     BEHAVIOUR
        rec X. prod<5,10>; ins<0, *>; X
     ENDPROC
     PROCESS Consumer [ extr, cons ];
     BEHAVIOUR
        rec X. extr<0, *>; cons<15, 20>; X
     ENDPROC
     PROCESS Buffer [ ins, extr ];
     BEHAVIOUR
        -- Buffer with capacity 5, as we do not include data,
        -- we can model it as 5 single cells in parallel.
        Cell [] Cell [] Cell [] Cell [] Cell
     ENDPROC
     PROCESS Cell [ ins, extr ];
     BEHAVIOUR
       rec X. ins<0, *>; extr<0, *>; X
     ENDPROC
ENDSPEC
```

Figure 3: Specification file for the P/C system

2 Description of timed-arc Petri nets

In order to simplify the definitions that will appear from now on, all arcs are supposed to have weight 1, but the extension to arcs with greater weight is quite straightforward.

Timed-arc Petri nets are an extension of Petri nets, in which tokens have an age associated (a real value), indicating the elapsed time from its creation, and the arcs connecting places with transitions have associated a time interval, which limits the age of the tokens to be consumed to fire the adjacent transition. In the particular model we consider, some transitions can be urgent, in the sense that no time can elapse once they are enabled.

Definition 1 *(Labelled timed-arc Petri nets)*

Given a finite set of visible actions Act, a labelled timed-arc Petri net (LTAPN) is a tuple N=(P, T, F, $times$, λ), where:

P is a finite set of places.

T is a finite set of transitions.

$(P \cap T = \emptyset)$ is fulfilled.

F is the flow relation $(F \subseteq (P \times T) \cup (T \times P))$

$times$ is a function that associates a closed time interval to each arc (p,t) in F, i.e.:

$times$: $F|_{P\times T} \rightarrow \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$

λ: $T \rightarrow Act \cup \{\tau\}$, is the labelling function. Thus, transitions can be labelled either with visible actions or with τ. These internal transitions (those labelled with τ) are considered urgent, i.e., no time can elapse when they are enabled, and they must be executed inmediately, unless they are in conflict with some other transitions. Intuitively, these τ-transitions correspond to actions that the system can make internally, and therefore, once they are enabled it does not need to wait to execute these actions.

As previously mentioned, tokens are annotated with real values, so markings are defined by means of multisets on $\mathbb{R}_0^+$. More precisely, a marking M is a function:

$$M : P \rightarrow \mathfrak{B}(\mathbb{R}_0^+)$$

Where $\mathfrak{B}(\mathbb{R}_0^+)$ denotes the set of finite multisets of non-negative real numbers. Thus, as usual, each place is annotated with a certain number of tokens, but each one of them has associated a non-negative real number (its age).

We will denote the set of markings of N by $\mathcal{M}(N)$, and using classical set notation, we will denote the number of tokens on a place p by $|M(p)|$.

As initial markings we only allow markings M such that for all p in P, and any $x > 0$ we have $M(p)(x) = 0$ (i.e., the initial age of any token is 0).

Definition 2 *(Marked labelled timed-arc Petri net)*

We define a marked labelled timed-arc Petri net (MLTAPN) as a pair (N,M), where N is a labelled timed-arc Petri net, and M is an initial marking of it.

Definition 3 *(Initial marking)*

Given a MLTAPN (N,M) we define the set of initial markings in the following way:

$$Init(N) = \{p \in P |\ |M(p)| > 0\}$$

As usual, from the initial marking we will obtain new markings, as the net evolves, either by firing transitions, or by time elapsing. A labelled timed-arc Petri net with an arbitrary marking can be graphically represented by extending the usual representation of P/T nets with the corresponding time information. In particular we will use the age of each token to represent it. Therefore, MLTAPNs have initially a finite collection of zero values labelling each place.

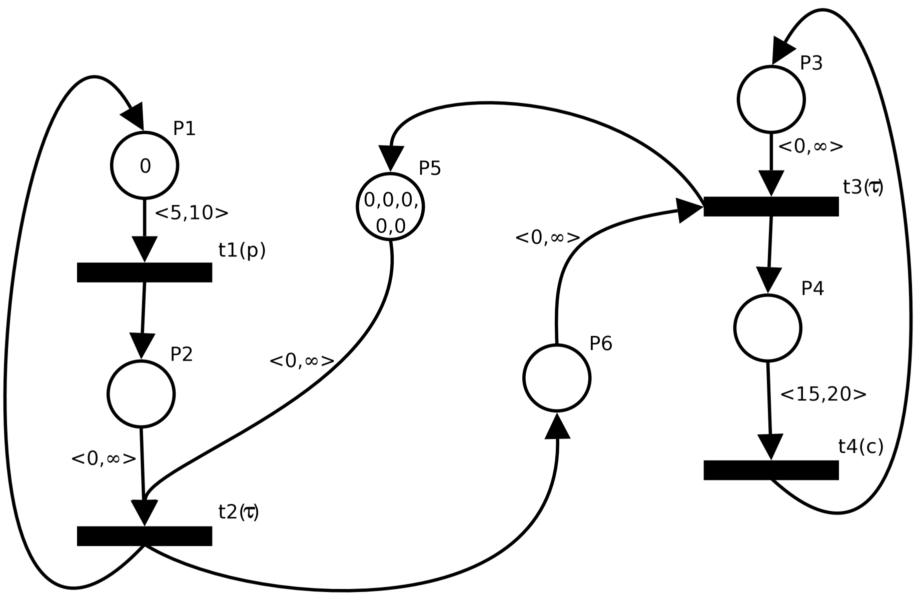

Figure 4: Timed-arc Petri net modelling the producer/consumer system

In Figure 4 we show a MLTAPN modelling a producer/consumer system. In it, transition t_1 represents the action corresponding to the manufacturing process of the producer (p), which takes between 5 and 10 units of time. Transition t_2 represents the action of including the generated object in the buffer (which is urgent). On the other hand, transition t_3 models the (urgent) action of taking out an object from the buffer, and transition t_4 models the processing that makes the consumer for the objects extracted from the buffer (c), which takes between 15 and 20 units of time. Initial tokens on p_5 represent the capacity of the buffer, in this case 5, whereas tokens on p_6 represent the objets in the buffer which has not been consumed yet. Finally, we can observe that the arc connecting the place p_5 with the transition t_2 is labelled by the interval $< 0\ ,\ \infty >$, because these tokens can be consumed at any time.

Let us observe that if the enabling time for the firing of one of these transitions (t_1 or t_4) expires, the system becomes deadlocked, because we obtain dead tokens either on p_1 or p_4.

Definition 4 *(Firing rule of a MLTAPN)*

Let N=(P,T,F,$times$,λ) be a LTAPN, M a marking on it, and $t \in T$:

1. We say that t is enabled at the marking M if and only if:

$$\forall p \in {}^\bullet t \ \exists\ x_p \in \mathbb{R}_0^+ \text{ such that } M(p)(x_p) > 0 \wedge x_p \in times(p,t)$$

   i.e., on each precondition of t we have some token whose age belongs to $times$(p,t). As usual, this will be denoted by $M[t\rangle$.

2. If t is enabled at M, it can be fired, and by its firing we reach a marking M' as follows:

$$M'(p) = M(p) - C^-(p,t) + C^+(t,p), \forall p \in P$$

   Where both the subtraction and the addition operators work on multisets, and:

$$C^-(p,t) = \begin{cases} \{X_p\} & \text{if } p \in {}^\bullet t, x_p \in times(p,t), \text{and } M(p)(x_p) > 0 \\ \emptyset & \text{otherwise} \end{cases}$$

$$C^+(t,p) = \begin{cases} \emptyset & \text{if } p \notin t^\bullet \\ \{0\} & \text{otherwise} \end{cases}$$

   Thus, from each precondition place of t we remove a token fulfilling 1, and we add a new token (with age 0) on each postcondition place of t. We denote these evolutions by $M[t\rangle M'$. It is important to stand out that these evolutions are in general non-deterministic, because when we fire a transition t, some of its precondition places could hold several tokens with different ages that could be used to fire it. Besides, we see that the firing of transitions does not take up any time. Therefore to model the passage of time we need the function *age* (defined below). By applying it we age all the tokens of the net by the same time:

3. The function *age*: $M(N) \times \mathbb{R}_0^+ \to \mathcal{M}(N)$ is defined by:

$$age(M,x)(p)(y) = \begin{cases} M(p)(y-x) & \text{if } y \geq x \\ 0 & \text{otherwise} \end{cases}$$

A marking net (N,M) will allow to elapse x units of time if and only if there is no τ-labelled transitions enabled at $age(M,y)$, $\forall y \in [0,x]$.

In this case, the marking obtained from M after x units of time without firing any transitions will be that given by $age(M,x)$.

Although we have defined the evolution by firing single transitions, this can be easily extended to the firing of *steps* or *bags* of transitions. Those transitions that could be fired together in a single step could be also fired in sequence in any order, since no *aging* is produced by the firing of transitions. In this way, we obtain step transitions that we denote by $M[R\rangle M'$. Thus, we define a timed step semantics, alternating step transitions and time elapsing, by:

$$\sigma = M_0[R_1\rangle_{x_1} M_1 \ldots M_{n-1}[R_n\rangle_{x_n} M_n$$

where M_i are markings, R_i multisets of transitions and $x_i \in \mathbb{R}_0^+$, in such a way that:

$$M_i[R_{i+1}\rangle M'_{i+1} y M_{i+1} = age(M'_{i+1}, x_{i+1})$$

Note that we allow $x_i = 0$ in order to capture the execution in time zero of two causally related steps.

Given a MLTAPN (N,M_0), we define $[M_0\rangle$ as the set of reachable markings on N starting from M_0. We say that (N,M_0) is bounded if for all $p \in P$ there exists $n \in \mathbb{N}$ such that for all $M \in [M_0\rangle$ we have $|M(p)| \leq n$.

Definition 5 *(Dead tokens)*

Let (N,M) be a MLTAPN. A token on a place p at the marking M is dead if it can never be used to fire any transitions, i.e., it will remain on p forever, just growing up.

Thus we say that a marking is dead when either it is the nullo marking or all its tokens are dead.

Definition 6 *(Safeness)*

We say that (N,M_0) is safe if $|M(p)| \leq 1$, $\forall p \in P$ and $\forall M \in [M_0\rangle$. We will say that it is soft-safe if for every reachable marking we have on each place at most one non-dead token.

2.1 Representation in TPAL

TPAL tool allow us to translate an algebraic specification written in TPAL to a marked labelled timed-arc Petri net (MLTAPN). For that, it generates a C structure called MTAPN, which has the required data to show the Petri net in a textual way. Then, another structure, called Red, is generated, which will contain the additional data that we need to draw it graphically. More specifically, it contains the coordinates of each element and some data to accelerate the redrawing and the simulation.

The C definition of these data structures follows:

```
struct Red{
      struct Red_Lugar *vector_lugares;
      struct Red_Transicion *vector_transiciones;
      struct Red_Arco_TP *lista_tp;
      struct Red_Arco_PT *lista_pt;};


struct Red_Lugar{
      double x,y;
      int marcas;
      int red;};

struct Red_Transicion{
      double x,y;
      BOOLEAN activada;
      int red;};

struct Red_Arco_TP{
      unsigned long t; /*source*/
      unsigned long p; /*destination*/
      unsigned int peso;
      struct Red_Arco_TP *sig;};

struct Red_Arco_PT{
      unsigned long p; /*source*/
```

```
unsigned long t; /*destination*/
unsigned int peso;
double t1,t2;
struct Red_Arco_PT *sig;};
```

Notice that we use arrays to keep the information about places and transitions, since its number is already known at that moment. Observe that we do not need to annotate its number, because it is known by its position in the array.

We can also observe we do not need to include the coordinates, since they are obtained from the adjacent elements.

2.2 Representation optimizations

As our first goal was to obtain a runnable version of the tool, we did not try to obtain a compact representation of the nets. Then, compresion techniques have not been used to represent the net structure or markings. Thus, it is obvious that the current version is quite limited in its use, as markings consume much space than needed, so our inmediate future work will focus on this point, reducing the space needed in the representation, and concretely on markings. Thus, we will use compression techniques for encoding the reachable markings of the generated nets, in order to deal with nets of hundreds or even thousands of places and transitions.

3 Description and use of the browser

Timed-arc Petri nets will be drawn in a window whose form is shown in Figure 5. It mainly contains the area where Petri net is pictured, and buttons to carry out the simulation (this functionality will be described in the following section).

The net is pictured in the top-left part of the window. This browser also allow to interact with the different elements of the net, obtaining information of them, and moving them with the mouse.

The numbers, written down in each place, show us the number of tokens that each place has. The labels of each transition are written in the center of these, and for a transition labelled with an internal action τ, "_tau_" is used. Further on we will see that it is possible to configure if the weight and the time intervals associated to the arcs are shown in the browser. These will be written down in the middle of the corresponding arc.

To move a place or a transition we click with the left mouse button on that place or transition, and we keep the button clicked while we drag it to the desired place, which

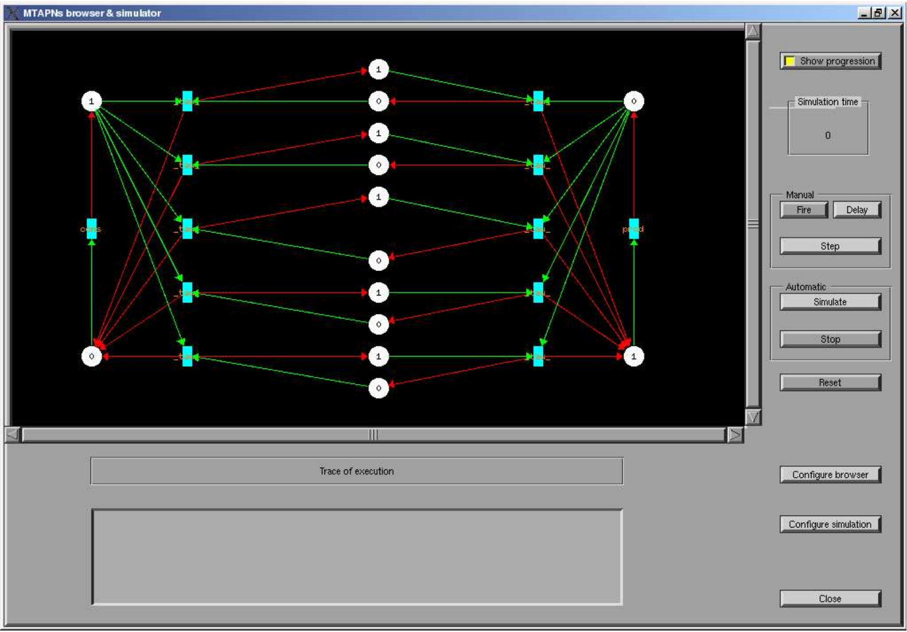

Figure 5: Timed-arc Petri nets viewer of TPAL

could be outside the visible area. We can move the canvas to see other parts of the painted net, with the scrollbars located at the bottom and on the right of the canvas.

To obtain a more detailed information on a concrete net element, we can click with the right mouse button on a place, a transition, a place-transition arc, or a transition-place arc, which generates a window with this information, as shown in Figure 6, for a place.

It is necessary to emphasize that this information appears with the colours that previously were configured, that is to say, if the transitions have as assigned colour the "cyan"(by default it is), then the information referring to each transition, will be painted in "cyan".

The window can be resized, as the resolution of the screen allows, and therefore a bigger portion of the net can be seen in that way.

3.1 Configuration window

The form of the configuration window is that shown in Figure 7.

There we can see 6 differentiated sections, and the buttons "Ok", "Cancel" and "Default". The sections are:

Places: This section contains the place configuration:

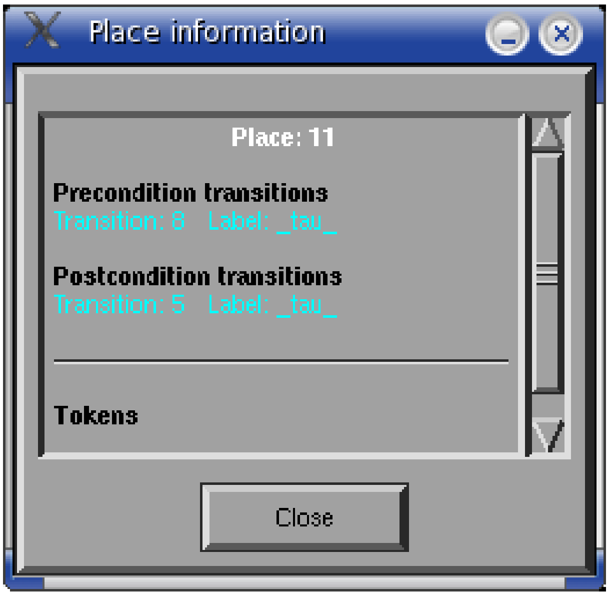

Figure 6: Place detailed information window

- Radius size (in pixels).
- Colour (from 0 to 32).
- Colour of the number (from 0 to 32) that appears in the center of the place, indicating the number of tokens that contains.

Transitions: Here we can configure:

- Transition height and width (in pixels).
- Colour (from 0 to 32).
- Enabled transition colour (from 0 to 32).
- Colour (from 0 to 32) of a manually selected transition to be fired.
- Transition label colour (from 0 to 32).

Place->Transition arcs: Here we can configure the place-transition arcs:

- Line width (from 1 to 4, measured in pixels).
- Colour (from 0 to 32).
- If the time interval must be shown or not.
- If the arc weight must be shown or not.

Transition->Place arcs: With respect to these arcs we can configure:

- Line width (from 1 to 4, measured in pixels).

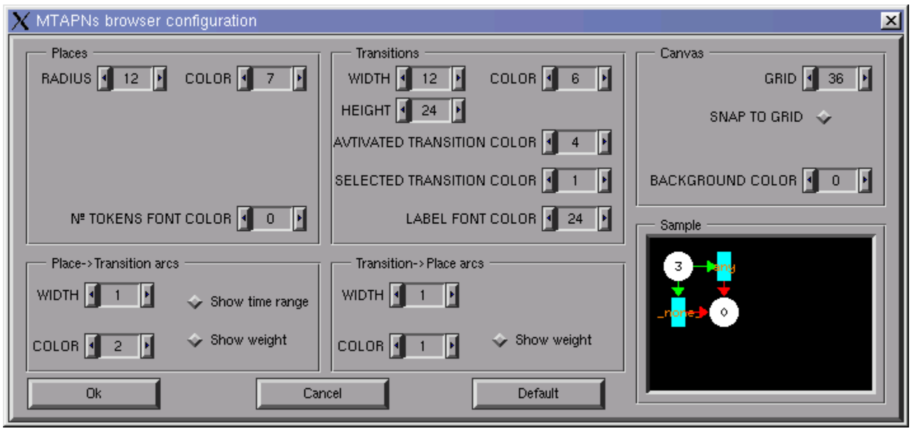

Figure 7: Browser configuration window

- Colour (from 0 to 32).
- If the arc weight must be shown or not.

Canvas: Here we can configure:

- The virtual grid width (in pixels), where the net places and the transitions will be drawn.
- If when moving a place or transition, it must be positioned or not in the positions defined by the grid.
- Background colour (from 0 to 32).

Sample: It is a small screen in which we can show the configuration results in order to see if it is satisfactory or not.

The buttons at the window bottom offer the typical functionality in this type of interfaces:

"Ok" Button: With this button the window is closed, accepting the changes. Then, the changes are applied to the Petri net shown in the main window.

"Cancel" Button: With this button the window is closed, discarding the changes.

"Default" Button: With this button all the default values are recovered, but the window is not closed.

4 Description and use of the simulator

The interface elements (shown in Figure 5), which drive the simulation are the following:

"Show progression": This button has 2 states. By default it is activated (the left box of the button is illuminated). Whenever it is pressed it changes from activated to deactivated and vice versa. When it is activated it means that the net has to be updated graphically to reflect the changes at each simulation step. On the contrary, if it is deactivated, these changes will not be reflected graphically in the net, but we obtain a greater speed in simulation.

"Fire": This button allow us to fire a transition, if there is any enabled. Otherwise this button will be deactivated. If an automatic simulation is in execution, this button will be deactivated.

"Delay": This button allow us to elapse an specific time in the net, if it is possible. Otherwise this button will be deactivated. After pressing this button, a window will be shown (Figure 8), indicating the maximum time that we can elapse, allowing to input a value between the minimum and the maximum possible times, although we can even cancel the process. If an automatic simulation is in execution, this button will be deactivated.

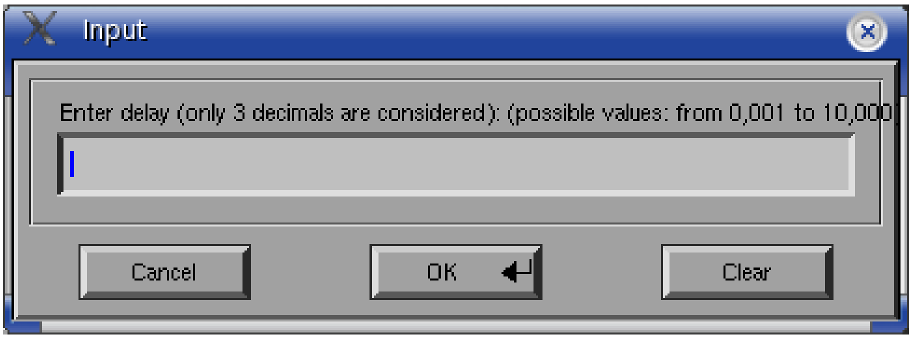

Figure 8: Window that allows to elapse manually an specific time in the net

"Step": The function of this button is to execute a single simulation step. If an automatic simulation is in execution, this button will be deactivated.

"Simulate": This button begins the automatic execution of the simulation. As a result of this process, all buttons are deactivated, except "Stop", and the name is modified to "Simulating . . . ". If the simulation has begun, either manually or automatically and then "Stop" is pressed, this button changes its label to "Continue".

"Stop": This button allows us to stop the simulation. If an automatic simulation is not in execution, this button will be deactivated.

"Reset": This button allow us to reset the net to the initial marking, erasing the data shown in “Trace of execution” and setting the simulation time to zero. This button remains deactivated until the simulation begins.

The visualization of the results obtained at each execution step is shown graphically in the browser, and textually in a zone of the window, right under the browser, titled “Trace of execution”. In addition, we can know the total time elapsed in the net in a zone titled “Simulation time”, located at the top-right.

Finally, we can also select the next transition to be fired using the central mouse button (of course, it must be enabled).

4.1 Configuration window

With this window (Figure 9) we configure:

Seed: It is the seed that will be used for the pseudo-randomness in the simulation process.

Avoid time locks: This is a radio button. Pressing it we try to avoid the deadlocks that come for the passage of time. These deadlocks appear when some tokens expire, and they do not allow the firing of some transitions any more.

Maximum simulation time . . . u.t.: This field defines the maximum time for the simulation, when using automatic simulation. It is used if the option *Complete simulation* has been selcted.

Complete simulation: If activated, the simulation will only stop if it exceeds the time defined by *Maximum simulation time*, or by pressing the “Stop” button in the main window of the viewer.

Simulate . . . steps: It is an alternative to the complete simulation, which causes the automatic simulation to conclude when a certain number of simulation steps have been made.

Simulation speed: Through a “slider” (element of the XForms GUI), moving the bar to the left or to the right, we can decrease or increase the speed with which the automatic simulation will be executed.

The buttons of the bottom part, which have the previously explained functionality.

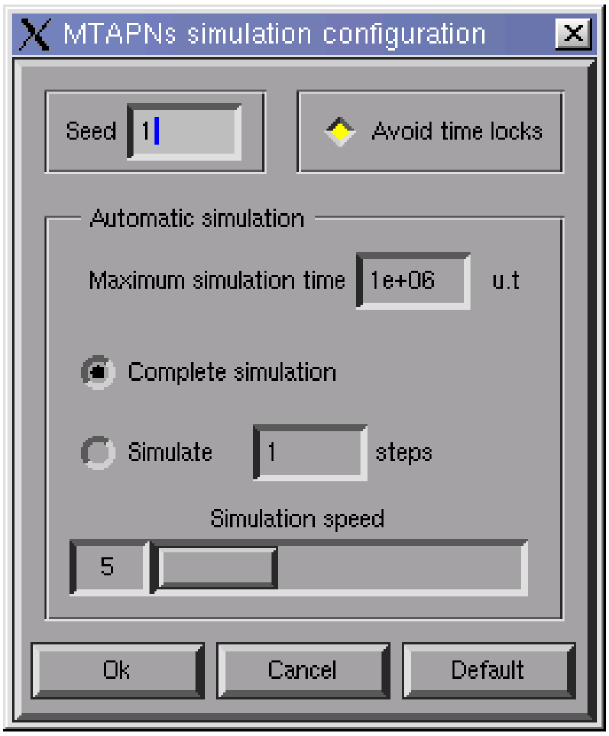

Figure 9: Simulator configuration window

5 Future works

According to the description that we have made, the tool has increased its functionality significantly. Our future work on it will include the following aspects:

- Implementation of verification properties algorithms, by using the timed-arc Petri nets.
- Translation either from the obtained graphs, or directly from the specifications to timed automata, and also to timed-probabilistic automata.
- Analysis and verification of the systems by using of model-checking techniques.
- Creation and edition of timed-arc Petri nets directly using a graphical environment.

References

[AN01] Parosh A. Abdulla and Aletta Nylen. Timed petri nets an bqos. In *Proc 22nd International Conference on Theory and Application of Petri Nets.*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70, 2001.

[BLPY95] J. Bengtsson, K. Larsen, P. Peterson, and Wang Yi. Uppaal.- a tool suite for automatic verfication of real-time systems. In *Proc of 4th Dimacs Workshop on Verification and Control of Hybrid Systems*, New Jersey, October 1995.

[BLT90] T. Bolognesi, F. Lucidi, and S. Trigila. From timed petri nets to timed lotos. In *Proceedings of Tenth International IFIP WG6.1 Simposium on Protocol Specification Testing and Verification*. North-Hollan, 1990.

[Bow96] Fred D. J. Bowden. Modelling time in petri nets. In *Proc. Second Australia-Japan Workshop pm Stochastic Models*, 1996.

[dFVM00] D. de Frutos, V. Valero, and O. Marroquín. Decidability of properties of timed-arc petri nets. In Ten H. Lai, editor, *Proceedings of 21st Int. Conference on Application and Theory of Petri Nets LNCS no. 1825*, pages 187–206, Aarhus, Denmark, 2000.

[DIT95] DIT UPM. *Lotos LAboratory. User Manual (Version 3R6)*, February 1995.

[Han93] Hans-Michael Hanisch. Analysis of place/transition nets with timed-arcs and its aplication to batchj process control. In *Application and Theory of Petri Nets*, volume 691 of *Lecture Notes in Computer Science*, pages 282–299, 1993.

[PMVR99] J.J. Pardo Mateo and Valentín Valero Ruiz. Prototipo de una herramienta de ejecucion de especificaciones en tpal. Technical Report DIAB-98-01-05, Universidad de Castilla-La Mancha, 1999.

[PV01] J.J. Pardo and Valentín Valero. Herramienta de ayuda al diseo y evaluación de sistemas concurrentes. Technical Report DIAB-01-01-14, UCLM, 2001. http://www.info-ab.uclm.es/sec-ab/Tecrep/diab010114.ps.

[PVCC01] J. Pardo, V. Valero, F. Cuartero, and D. Cazorla. Automatic translation of tpal specification into dynamic state graphs. In *Proc of APSEC'01*. IEEE Computer Society Press, 2001.

[PVR01] J.J. Pardo, Valentín Valero, and M.Carmen Ruiz. Tpal: Una herramienta de simulación de sistemas concurrentes con restricciones temporales. In *Sistemas Distribuidos: Modelos y Aplicaciones*, pages 157–169. Albacete(Spain), July 2001.

[PVRC02] J.J. Pardo, Valentín Valero, M Carmen Ruiz, and M Emilia Cambronero. New features of tpal for the specification and analisis of concurrent systems. Technical Report DIAB-02-01-25, UCLM, 2002. http://www.info-ab.uclm.es/sec-ab/Tecrep/diab020125.ps.zip.

[UL04] Javier Ulibarri López. Simulador de redes de petri con arcos temporizados e integración en tpal. Master's thesis, Universidad de Castilla-La Mancha, Julio 2004.

[vdA93] W.M.P. van der Aalst. Interval timed coloured petri nets and their analysis. *Lecture Notes in computer Science*, 691:451–472, 1993.

[VdFC99] V. Valero, D. de Frutos, and F. Cuartero. On Non-Decidability of reachability for Time Arcs Petri Nets. In *Proc. 8th Workshop on Petri Nets and Performance Models, PNPM'99*, pages 188–196, 1999.

[VPC02] Valentín Valero, J.J. Pardo, and Fernando Cuartero. Translating tpal specifications into timed-arc petri nets. In *Proceedings of ICTAPN'2002*, Adelaine, Australia, June 2002.