

Chapter 2

Getting started

After *GreatSPN2.0.2* has been installed and the user environment has been set up according to the directions given in the Appendix C, the user can start the package and build and analyze his/her models. The aim of this chapter is to quickly introduce the user to (modeling) using *GreatSPN2.0.2*. By following this tutorial, the user will be able to construct a Petri Net model and to analyze it by means of the *GreatSPN2.0.2* graphical interface. The reader interested in a more in-depth presentation of the various *GreatSPN2.0.2* features may refer to Chapter 3. Throughout this chapter we shall use as an example the model of the well-known multiple-reader-single-writer problem in the access of a shared data base.

2.1 The Readers–Writers GSPN model

Let us consider a set of “processes” concurrently accessing a shared data base. When a process issues an access request, it declares whether a read or a write operation is required. Read operations may proceed concurrently with each other, while write operations require an exclusive access in order to maintain the consistency of the data base. A GSPN model of this system is depicted in Figure 2.1. A token in place *think* represents a process performing some local activity. After a random amount of time, a data base access request is issued (timed transition *arrival* fires). The type of request (read or write) is randomly chosen with equal probability (immediate transitions *isread* and *iswrite* have the same weight). If the operation is a read, then the access is granted if no other process is performing a write operation on the data base (place *nowrite* is marked), otherwise the process waits in place *Rqueue* until the access can be performed safely. The beginning of a read operation is represented by the firing of transition *StartR*, that has two effects. First, place *nowrite* is marked, meaning that other readers can access the data base. Second, place *reading* is marked, forbidding therefore the access to any writer process (there is an inhibitor arc from place *reading* to transition *StartW*). After the firing of the timed transition *EndR*, the status of the process is reset to “thinking”.

Conversely, if the requested operation is a write, the process waits (in place *Wqueue*) until access can be granted, i.e. when no other process is performing a write access (place *nowrite* is marked) and no other process

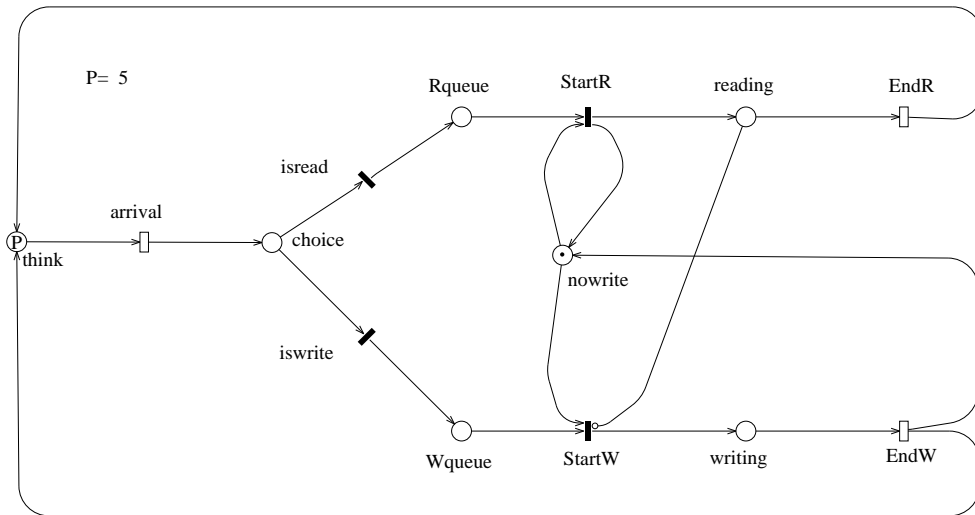


Figure 2.1: The Readers–Writers GSPN model

is performing a read access (place *reading* is empty). At the completion of the write operation, modeled by the firing of transition *EndW*, the absence of write access is signaled to other processes by marking place *nowrite*, and the status of the process is reset to “thinking”.

2.2 Starting GreatSPN

GreatSPN2.0.2 is started by typing `greatspn` on the command line and pressing <Return>. After a few seconds the Control Panel, depicted in Fig. 2.2, pops up and the user can start a work session.

WARNING! When *GreatSPN2.0.2* GUI is launched for the first time a window is displayed before the Control Panel pops-up in which it is asked to the user to fill in the corresponding areas if he/she desires to change the default setting for some environment variables (see chapt. 3 for a more detailed description of this window). The user has to press the “OK” button to confirm either the modification made in the window areas or the default settings: the information contained in this window are saved into the `$HOME/.greatspn` file.

To create a new model, the user has just to start creating places and transitions (as discussed in Section 2.3), while if the user desires to modify an already created model, he/she can retrieve it by using the **File** menu.

2.3 Creating the Readers–Writers model

To create (or to modify) a model, the user selects the *action* to be performed and the *object type* (e.g. places, immediate transitions, place markings, etc.) on which the action will be applied. The general principle of

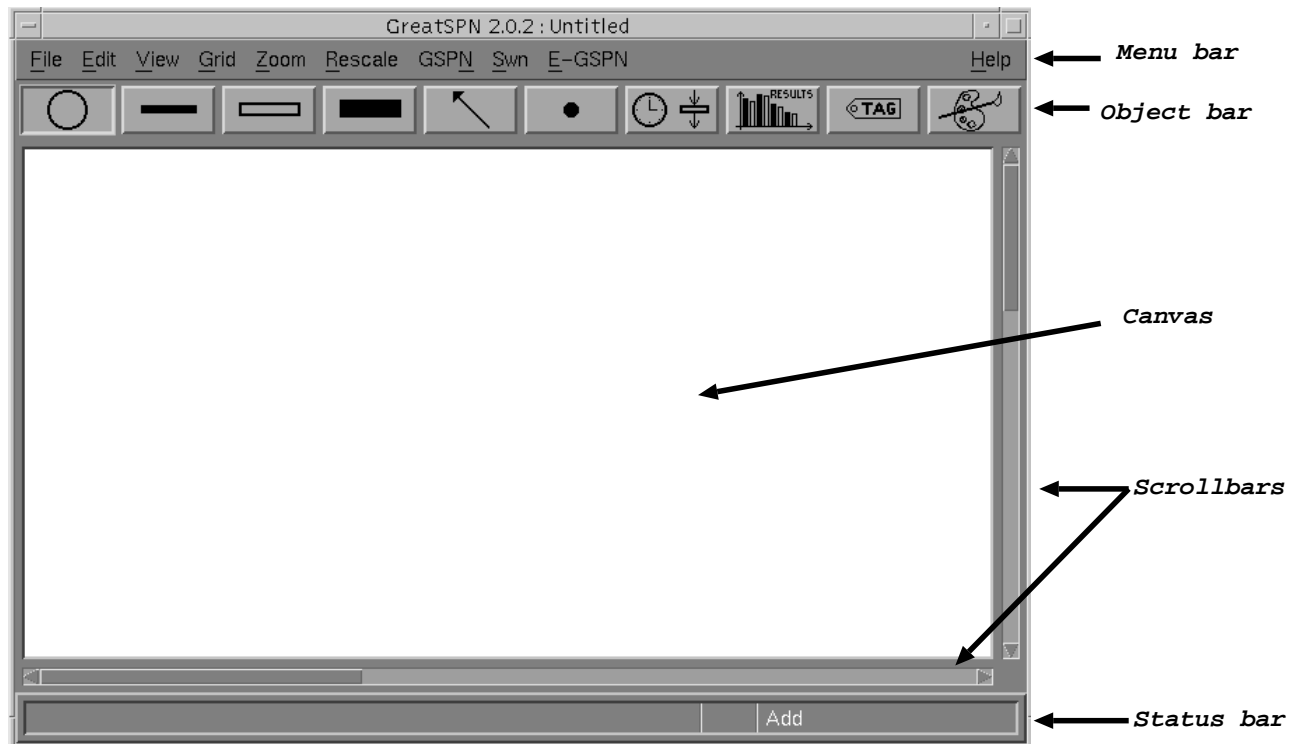


Figure 2.2: The *GreatSPN2.0.2* Control Panel

behavior of the *GreatSPN2.0.2* graphical interface is that the action selected by the user from the **Action** menu (shown in Fig. 2.3) becomes the default action until a new one is chosen, and such action affects only objects of the type currently selected. The current action is displayed in the *status bar* on the right. The **Action** pop-up menu is activated by pressing the right mouse button on any position of the *GreatSPN2.0.2* working area. To select an action, the user has to click with the left mouse button over the desired pull-down option¹. An object type is selected by moving the mouse cursor on one of the icons of the *objects bar* (see Figure 2.2) and by clicking over it with the left mouse button. The object currently selected remains highlighted until another object type is chosen. The *mouse help* window, activated by selecting **Help**→*Mouse Help*, displays the action associated with each mouse button. Let us start the creation of the Readers–Writers GSPN model by creating the set of places first. To create places, we set the type of object to “place” by selecting the *place* icon in the *object bar* (indicated by a circle), and after this operation the shape of the mouse cursor is changed into a circle. After the selection of the **Action**→*Add* option, we can move the mouse cursor within the working area and start laying down the places of the net by clicking the left mouse button in the proper position, in order to obtain the screen image shown in Figure 2.4. The *GreatSPN2.0.2* graphical interface displays only a window on the real working area, that can be scrolled in all the directions by using the *scrollbars* placed on the right and bottom sides of the Control Panel.

¹ In the rest of this manual we shall use the notation **Menu Item**→*Option* to denote an option in a specific menu item of the *menu bar*. For example, **Action**→*Create* denotes the *Create* option of the **Action** menu.

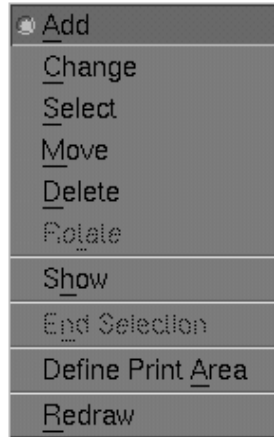


Figure 2.3: The *GreatSPN2.0.2* action menu

To create the transitions we proceed as for places, that is by first choosing one of the three transition icons available, i.e. deterministic (represented by a thick black box), exponential (represented by a white box), and immediate (represented by a thin black box), and then by creating them. Note that we don't need to select the **Action**→*Add* action again, since we didn't change the selection done when places were created. The screen situation after the addition of transitions looks like Figure 2.5. Note that we have created transitions having



Figure 2.4: Place layout of the Readers–Writers model of Fig. 2.1

Figure 2.5: Place and transition layout of the Readers–Writers model

different orientations. To change the orientation of a transition, we can use the middle mouse button (each time this button is clicked, the transition rotates of 45 degrees clockwise) before clicking the left button to actually create the transition itself.

After a place or a transition has been placed on the working area, it can be moved by selecting the **Action**→*Move*

option. An object can be dragged on the *canvas* by clicking on it with the left mouse button, by moving the cursor on its new position and by clicking the left button again.

The editor assigns default names (“tags”) to places (Px) and transitions (Tx for timed and tx for immediate) where x is an integer representing the objects creation order. Such tags are displayed if the **View**→*Tag* option has been selected. Tags overlapping other objects of the net can be moved around in the same way as places or transitions, that is by selecting the **Action**→*Move* option after clicking on the *tag* icon, and by clicking on the place or transition to which the tag is associated.

Now that we have created the nodes of our Petri net, we are ready to connect them with arcs. This can be accomplished by choosing the *arc* icon in the *object bar* (indicated by an upward arrow). Again, if we didn’t redefine the default action, we don’t need to re-select the **Action**→*Add* option. To create an input arc from place $P1$ to transition $T1$ (see Figure 2.6), we have to click the leftmost button of the mouse twice: first over place $P1$, and then over transition $T1$. The output arc connecting transition $T1$ to place $P2$ can be created by clicking the left mouse button first over $T1$, and then over $P2$. The inhibitor arc connecting place $P6$ to transition $t5$ is created

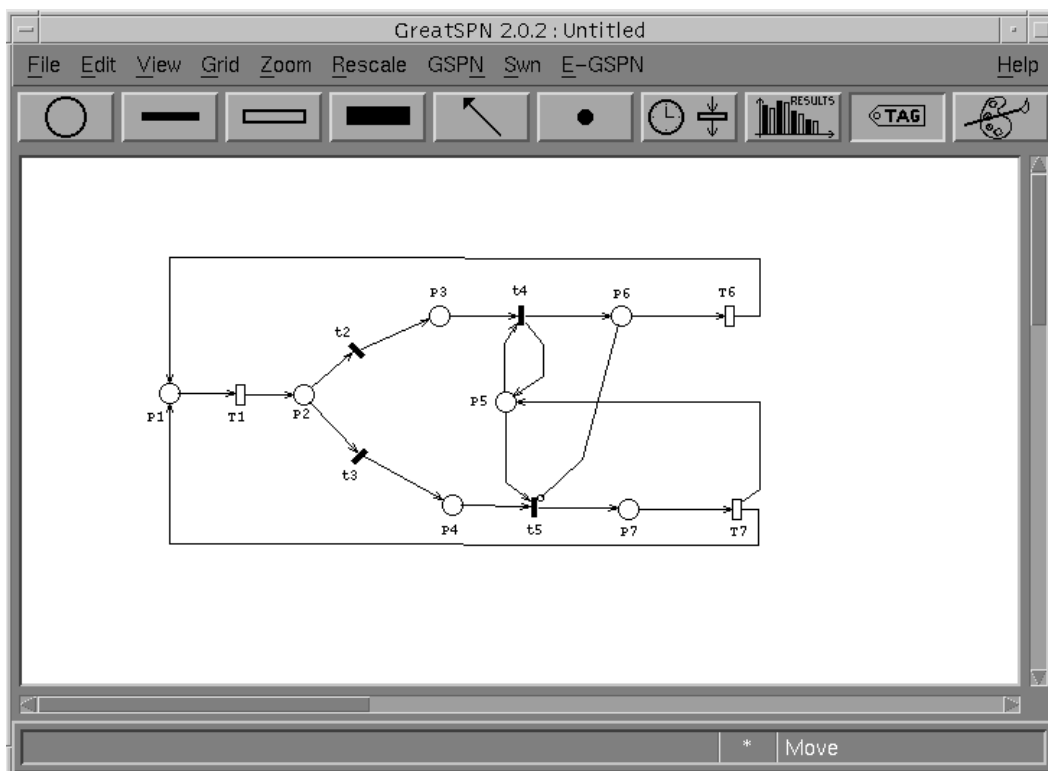


Figure 2.6: Creating arcs

by first clicking over $P6$ with the *middle* mouse button, and then over $t5$ with the left button. In the case of the output arc connecting transition $T7$ to place $P1$, for aesthetical reasons we want to put two intermediate points between the place and the transition, rather than connecting them with a straight line. Intermediate points are

added by just clicking the left mouse button over the desired position, provided that it is not too close to a node of the net.

In Petri nets it is forbidden to draw arcs connecting either places to places or transitions to transitions. *Great-SPN2.0.2* enforces this rule by forbidding the creation of such arcs. Note that once one has started the drawing of an arc, there is no way to interrupt the action, so if we realize that we started to draw an arc that we shouldn't, the only way to get out is to complete the arc and to delete it afterwards. To delete an arc (or, more generally, an instance of the currently-selected object), we have to click over the object we want to delete after the selection of the **Action**→*Delete* option.

Now the topology of the network is complete, and we may proceed defining the initial marking. Place marking can be defined either directly, by associating an integer number of tokens with the place, or by means of a rate parameter which has been previously defined. In either case, the association of an initial marking with a place is performed by clicking with the left mouse button over the place we want to consider after the **Action**→*Change* option and the *place* icon have been selected. To create a *marking parameter* the user has to click over the *token* icon (graphically represented as a black dot) and to select the **Action**→*Add* option. After that, the user has to move the cursor in an empty *canvas* region and click the left mouse button. A dialog box (see Fig. 2.7) will pop-up, and will ask us to enter the name of a marking parameter (in our case the name is "P") and the corresponding numerical value (a positive integer).

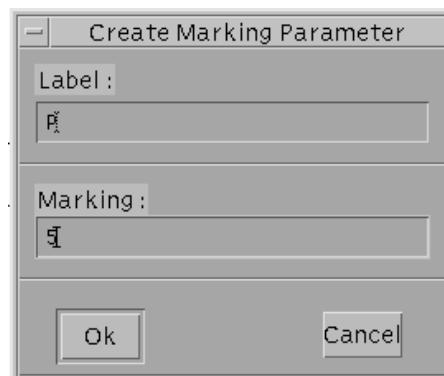


Figure 2.7: Dialog box for the creation of marking parameters

After clicking on the button "Ok" of the above dialog box, the definition of the marking parameter ("P=5") will appear on the chosen *canvas* position. Starting from this moment, the name "P" can be used as initial marking specification for any place in the net. In our example, two places hold a non-null initial marking, and we can define that by clicking the left mouse button on each of them. The *Change Place Properties* window (see Fig.2.8) will pop up, so that the place marking can be changed by specifying the initial value in the "Marking:" area.

Alternatively, the same action can be performed by selecting the *token* icon. The initial marking of a place can

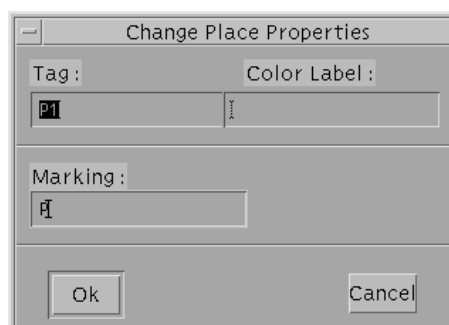


Figure 2.8: Dialog box for changing place properties.

be specified either as a nonnegative integer value or by using the name of an already-defined marking parameter.

As the network specification is complete, we may display a “nicer” version with rounded arcs by selecting the **View**→*Spline* option. This produces a net looking like the one depicted in Fig. 2.9.

So far we have retained the default names given by the editor at the moment of the object creation. Although this is useful to speed up the editing procedure, it may yield to a poor model readability, that can be improved by giving meaningful names to places and transitions. Tags can be modified by selecting the “tag” object, by choosing the **Action**→*Change* option and by pointing and clicking the left mouse button on the corresponding place or transition. A dialog box will pop up and will ask the user to specify the new object tag.

2.4 Saving and printing the model

After the model definition has been completed, we can save its description and/or print it using several different formats, by means of the **File** menu. A model is saved by selecting the **File**→*Save* option. If the model had been previously saved, this operation will cause the old description to be overwritten. If one wants to keep the old description, he/she can use the **File**→*Save As* option of the above menu to specify a new name for the net. The net description files are saved in the user directory defined by setting the environment variable `GSPN_NET_DIRECTORY` (as specified in the `$HOME/.greatspn` file of the user: see the Appendix C for details). *GreatSPN2.0.2* provides the possibility of specify comments, which are saved with the net description and which can be subsequently re-edited. To add a comment, simply select **File**→*Comment* option, and use the *Edit Net Comment* dialog box which pops-up. To create printouts of a model, or Encapsulated PostScript (EPS) files suitable for inclusion in \LaTeX documents, we have to select the **File**→*Print* option. This option affects only that portion of the model that is included in the currently-defined *print area*, that is displayed as surrounded by a dotted line if the **View**→*Print Area* option is selected (see Fig. 2.9). To define the print area, we have to:

1. select the **Action**→*Define Print Area* option (the cursor shape will be changed into a cross);

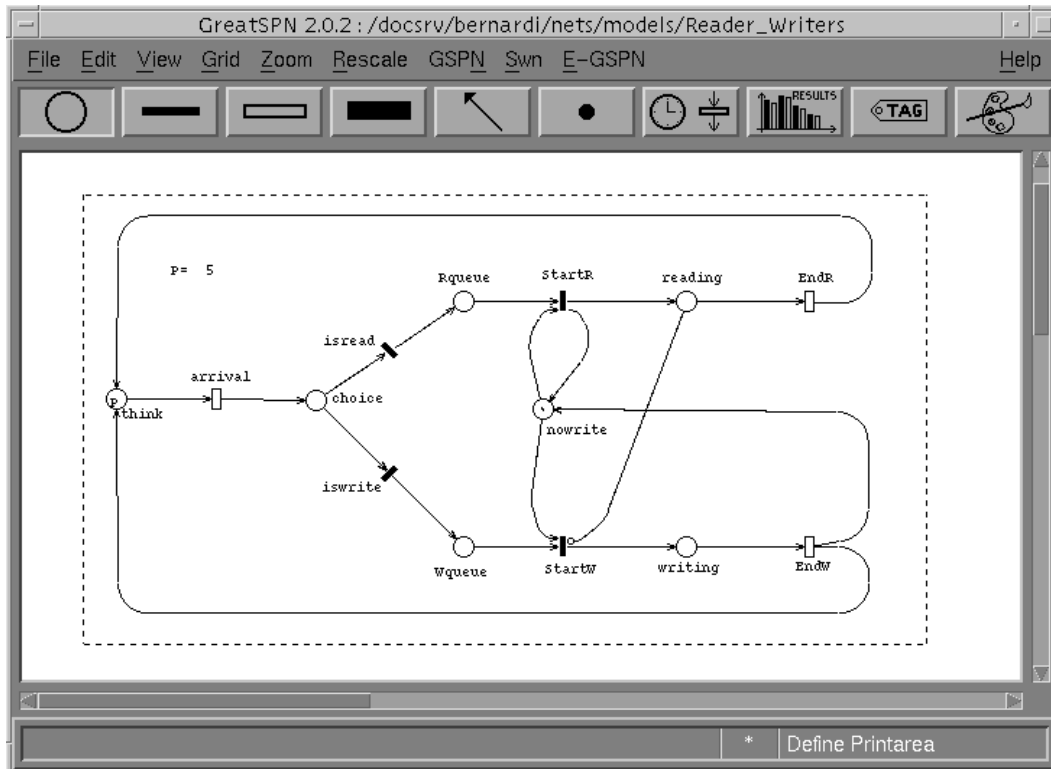


Figure 2.9: Print area used for the Readers-Writers model

2. click with the left mouse button over the *canvas* point corresponding to the upper-left corner of the print area;
3. move the mouse cursor on the lower-right corner of the desired print area and click either the left or the middle mouse button.

After the above operations have been completed, the **File**→*Print* command must be selected to cause the *Print* dialog box (shown in Fig. 2.10) to pop up. In the window contained in the leftmost part of the *Print* dialog box, *GreatSPN2.0.2* shows an overview of the entire working area (remember that only a window on a portion of the working area is shown in the Control Panel). The print area is surrounded by a thin black frame, that can be adjusted (allowing the redefinition of the print area) by positioning the mouse cursor over the lower-right corner (indicated by a small black square) and dragging it by clicking with the left mouse button *without releasing* the mouse button.

The six icons displayed just below the above window allow us to set:

- the format of the output , that can be chosen between raw PostScript (by clicking on the *PS* icon) and Encapsulated PostScript (by clicking on the *TeX* icon);

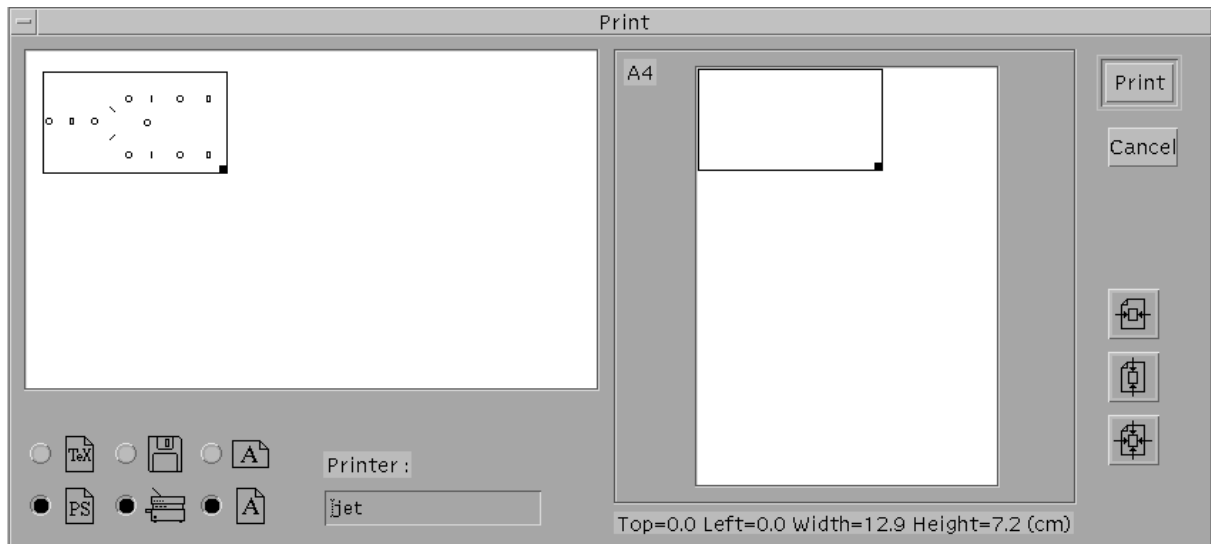


Figure 2.10: *GreatSPN2.0.2* Print dialog box

- the destination of the output, that can be either a file (*Diskette* icon) or a PostScript printer (*Printer* icon): if the EPS format is chosen then only a printout on a file is allowed;
- the orientation of the output, that can be either portrait or landscape (the two *A* icons).

The right part of the *Print* dialog box contains a window that shows the page layout (an A4 size sheet is assumed). The placement of the print area over the sheet can be changed either by dragging the thin black rectangle or by clicking over one of the three icons placed at the right of the window (that allow centering the picture on either dimension). Finally, the printout with the desired options is performed by clicking with the left mouse button on the “Print” button. If we decided to save the printout on a file, *GreatSPN2.0.2* will ask the user (by means of a suitable dialog box) to enter a file name that will be placed either in the default PostScript (PS) or Encapsulated Postscript (EPS) directory. The default PS and EPS directories can be set by means of the environment variables `GSPN_PS_DIRECTORY` and `GSPN_EPS_DIRECTORY` in the `$HOME/.greatspn` file (see Appendix C).

2.5 Analysis of the Readers-Writers model

Once we have defined both the net structure and the initial marking of the Readers–Writers model we can have a first understanding of its dynamic behavior by playing the “token game”. *GreatSPN2.0.2* provides the user of an interactive token game that can be started by selecting the **GSPN**→*Simulation...* option. The *Simulation* window pops up and all the transitions of the model that are enabled in the initial marking become blinking (see Fig. 2.11). To simulate a possible behavior of the modeled system we have simply to click with the leftmost

button of the mouse over the enabled (i.e. blinking) transition we want it to fire. In our running example, only transition *arrival* is enabled in the initial marking; by clicking on it, the firing action is executed, i.e. a token is removed from the input place *think* and a token is added to the output place *choice*. The new reached marking enables the two immediate transitions *isread*, *iswrite*: we can choose to fire one of them and to simulate the corresponding firing action (and so on). By default the “Untimed” and the “Forward” options of the *Simulation* window are set to play the forward token game: it is possible to play the backward token game by setting the “Backward” option in the *Simulation* dialog box.

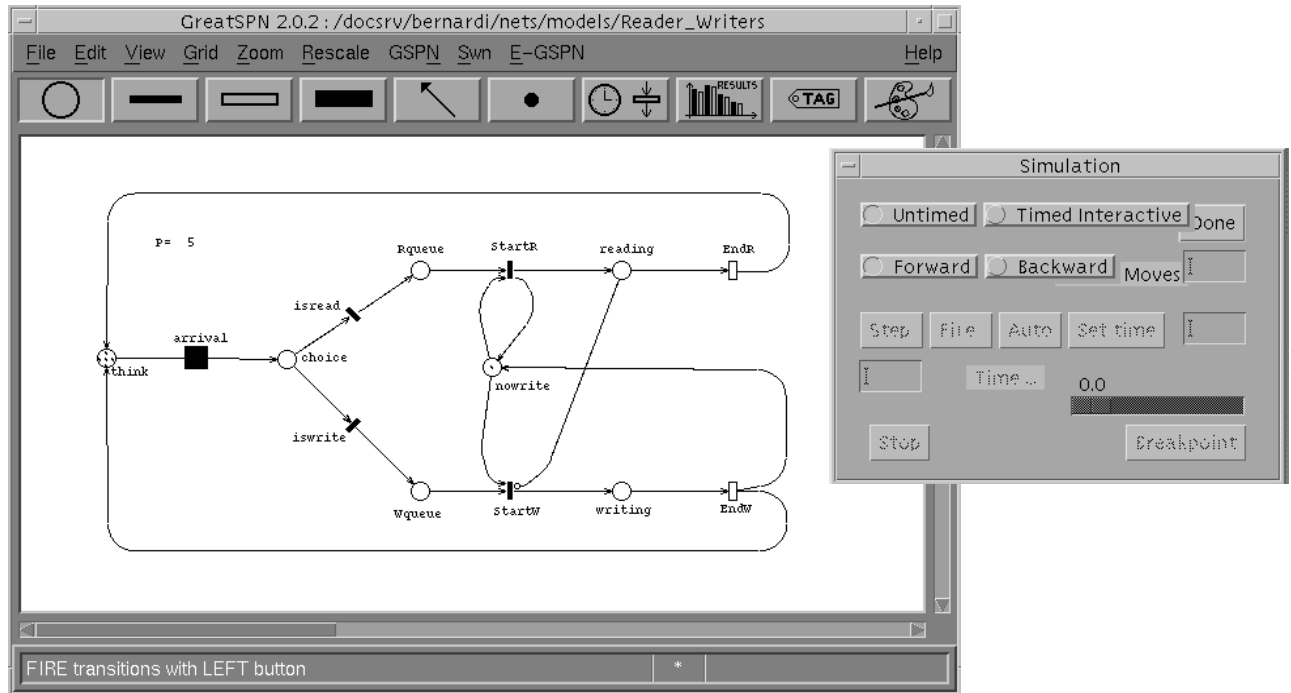


Figure 2.11: Token game of the Readers–Writers model.

GreatSPN2.0.2 provides the user with a set of structural analysis algorithms that can be used to validate the models. The user can access the above algorithms by selecting the **GSPN**→*Struct* option.² For example, the computation of minimal-support, canonical Place Invariants (by means of a modified Martinez–Silva algorithm [29]) can be accomplished by means of the option **GSPN**→*Struct*→*P invariants*. *GreatSPN2.0.2* will visualize the *Console* window (shown in Fig. 2.12), allowing one to start the P-invariant computation by clicking with the left mouse button over the “Start” button. At the end of the above computation, the *Console* window will contain the results, as displayed in Fig. 2.13.

After a GSPN model has been constructed and validated, it can be analyzed by means of the different perfor-

² **WARNING!** Before launching a *GreatSPN2.0.2* solver be sure that the hostname set in the “Hostname:” left area of the **File**→*Options* window is the name of the machine on which the Control Panel has been started.

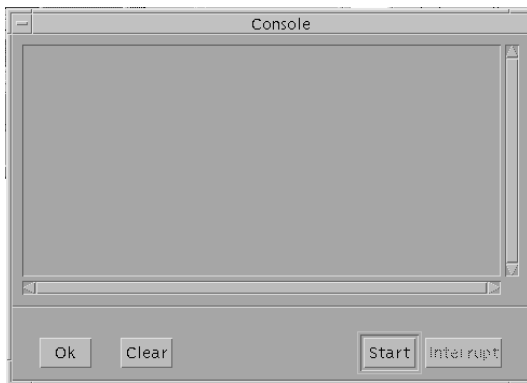


Figure 2.12: *GreatSPN2.0.2* Console

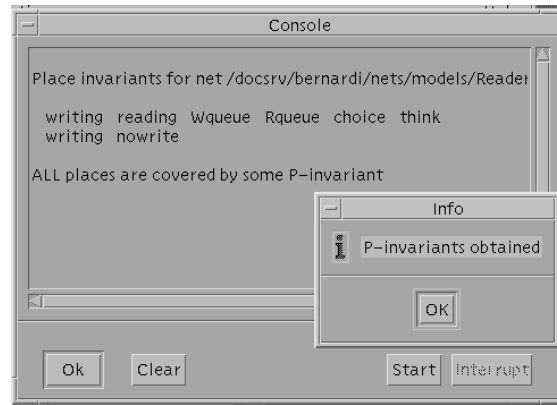


Figure 2.13: Results of P-invariant computation for the Readers-Writers model

mance evaluation techniques provided by *GreatSPN2.0.2*. Before starting the performance analysis of a given GSPN model, its performance-related parameters must be defined. The default rates associated with transitions can be displayed on the screen by selecting the **View**→*Rate* option. Rates overlapping some other object of the net can be moved around in the same way as tags, by selecting the **Action**→*Move* action after the icon corresponding to rates (indicated by a clock close to a timed transition) has been selected. Transition rates may be defined as positive real numbers, name of rate parameters, or marking-dependent expressions, governed by a context-free grammar (described in Appendix A). In this example we will only use rate parameter specifications, that are created much in the same way as marking parameter, by choosing the **Action**→*Add* action together with the *rate* icon. A dialog box (see Fig. 2.14) will pop up, allowing us to specify the name and the definition of rate parameters. We will define three rate parameters for transition rates: $arr = 1.0$, $rr = 2.0$, and $wr = 0.5$; and two rate parameters for choice probabilities between conflicting immediate transitions: $isr = 0.8$, and $isw = 0.2$. After their definition, the above parameters can be used to specify the transition rates (weights) by choosing



Figure 2.14: Dialog box for the creation of rate parameters

the **Action**→*Change* action with the exponential (immediate) transition type selected, by clicking with the left mouse button over the appropriate transitions and by filling the *Rate or Rate Parameter (Weight)* field of the corresponding *Change Transition Properties* window (see Fig.2.15).



Figure 2.15: Windows for defining/changing properties of timed (A) and immediate (B) transitions.

The same window, depicted in Fig.2.15(A), allows the specification of the enabling dependence. The default enabling dependence for timed transition is of the “infinite server” type, but it can be changed by choosing the appropriate option (*Infinite*, *Marking Dependent*, *1-Server*, and *Load Dependent*). In our example, only the *arrival* and the *endR* transitions are of the “infinite server” type, so the enabling dependence of all the other ones must be changed to *1-Server*. To change the priorities of immediate transitions, use the scrollbar on the bottom-left of the *Change Transition Properties* window (Fig.2.15(B)) of the corresponding transitions to increase/decrease their priorities.

With the specification of transition rates and probabilities as shown in Fig. 2.16, we have completed the specification of the behavior of the model. *GreatSPN2.0.2* provides three different performance analysis methods, namely computation of bounds for the throughput of transitions, Markovian solution and simulation. In this chapter we present an example of the Markovian solution of the Readers–Writers GSPN model, while the other techniques will be covered in Chapter 4. The Steady–State solution of the Embedded Markov Chain corresponding to the GSPN model of the Readers–Writers system of Fig. 2.16 is obtained by selecting the **GSPN**→*Solve*→*GSPN Solution*→*Steady State* option. The *Console* window will pop-up again, and after we click with the left button on the “Start” button, *GreatSPN2.0.2* will start the analysis phase. At the end of the analysis, *GreatSPN2.0.2*

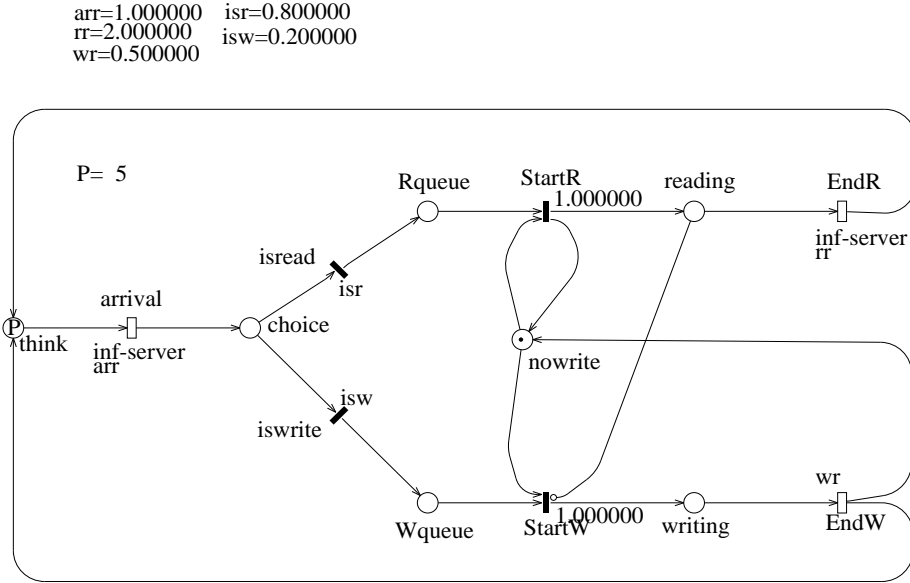


Figure 2.16: The Readers-Writers model with transition rate/probability specification

will show on the screen the values of the throughput of the various timed and immediate transitions, as well as the values of the defined performance indices (see Figure 2.17). We can visualize the distributions of token into places by selecting the **Action**→*Show* action together with the *result* icon, and by clicking on the place of interest. In Fig. 2.18 it is shown the token distribution for place *Wqueue*.

2.6 Colored version of the Readers-Writers model

Let us suppose that the processes which are concurrently accessing to the shared data base have different behaviors; in particular, a group of them access to the data base only to perform a write operation, while another group can issue either a writing or a reading request. In this section we describe how to obtain the colored *Great-SPN2.0.2* version of readers-writers model (see Fig.2.19) that captures the different behavior of the two kinds of processes. Starting from the previous non-colored model of the readers-writers system we first need to define the basic color classes representing the two kinds of processes. To create (or to modify) a basic color class definition simply click with the left mouse button on the *color* icon (indicated by a palette and a paintbrush) of the *object bar* and pop-up the **Action** menu, using the right mouse button, in order to select the “Add” option as the current action. After these operations, choose a place in the *canvas* to locate the definition of the class and click with the left mouse button: the *Create Color Definition* window pops-up (see Fig.2.20). The top left area named as “Label:” has to be filled with the name of the color class. The “Colorset” toggle, by default, is already switched on (a black dot is displayed in the circle near the toggle), indicating that the current definition is a definition of a

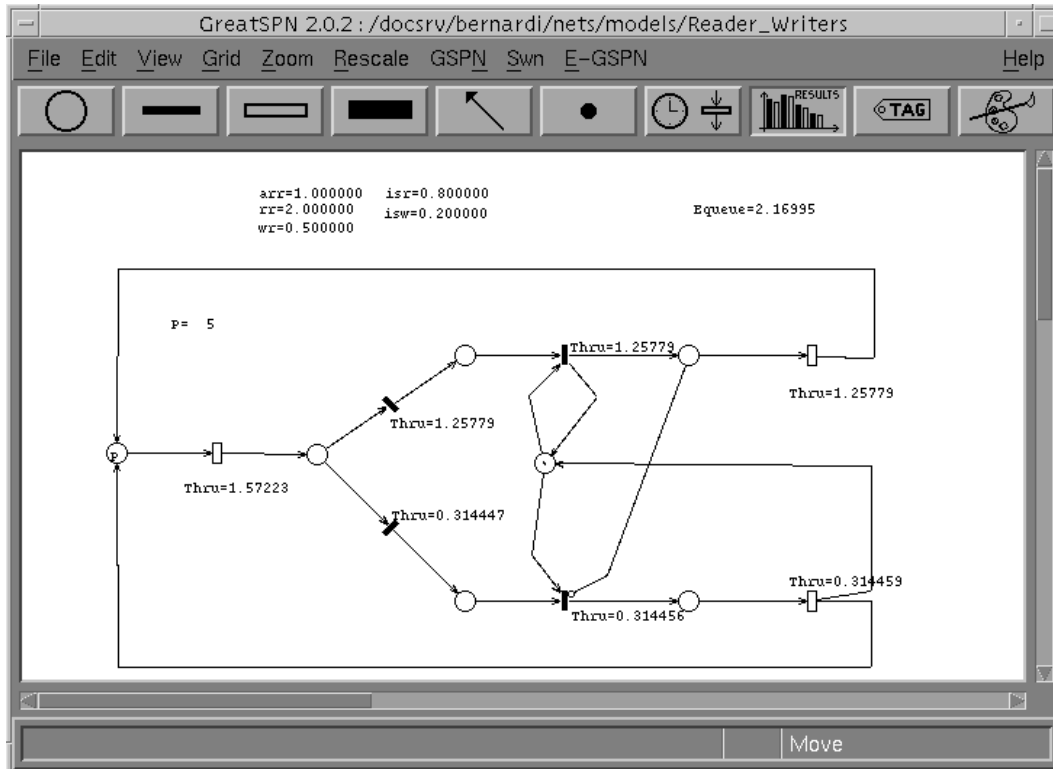


Figure 2.17: *GreatSPN2.0.2* canvas after the Readers–Writers GSPN model has been solved

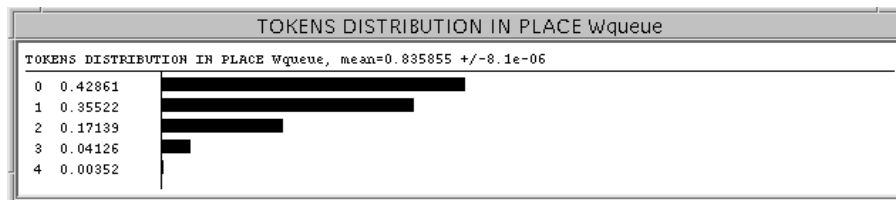


Figure 2.18: Token distribution in place *Wqueue*

basic color class. In the “Definition:” area, the definition of the basic color class is written using the SWN syntax (see Appendix A): in the example the class P is the unordered union of two static subclasses $P1$ and $P2$. These two colored subclasses have to be defined as well, following the same procedure described above for the color class definition, i.e., by recalling and filling the areas of the *Create Color Definition* window for each of them (see Fig.2.21). In our example, we have used two different alternatives of the SWN syntax to express the subsets of colors $P1$ and $P2$: the color subclass $P1$ is defined as the set of two elements $p1, p2$ while the color subclass $P2$ is a set of three elements $c1, c2, c3$. Once the basic color classes has been defined, we proceed as follows:

- add color domains to places that may contain colored tokens. To modify place attributes, click on the *place* icon and select from the **Action** menu the *Change* option; place color domain P has to be written on the

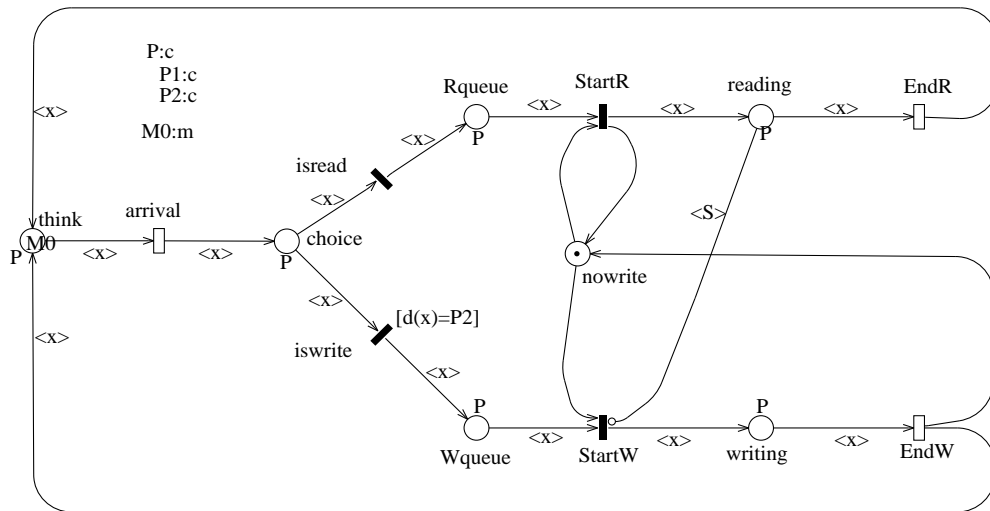


Figure 2.19: SWN model of the Readers-Writers system.

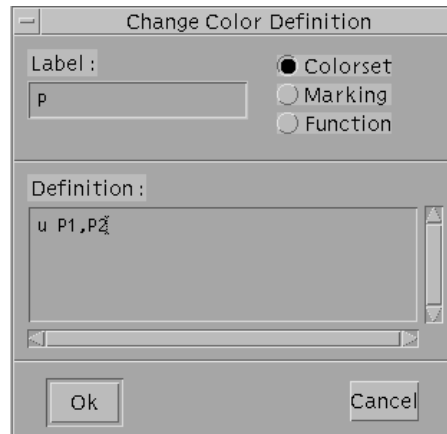


Figure 2.20: Create Color Definition window.

“Color label:” area of the *Change Place Properties* window (see Fig.2.8). In the model of Fig.2.19, all the places have color domains, except for place *nowrite*;

- add color attributes to the corresponding input/output arcs. To modify arc attributes, press the *arc* icon (we don’t need to select again the **Action**→*Change* option since it is the current action), and click on the interested arc with the left mouse button; the *Change Arc Properties* window of Fig.2.22 pops-up. Press the “Color” toggle to set the right area into “Color” mode and then add in the area the color function according to the SWN syntax. In the model of Fig.2.19, all the arcs are characterized by the identity function $\langle x \rangle$, except for the input/output arcs of the non colored place and for the inhibitor arc connecting place *reading*

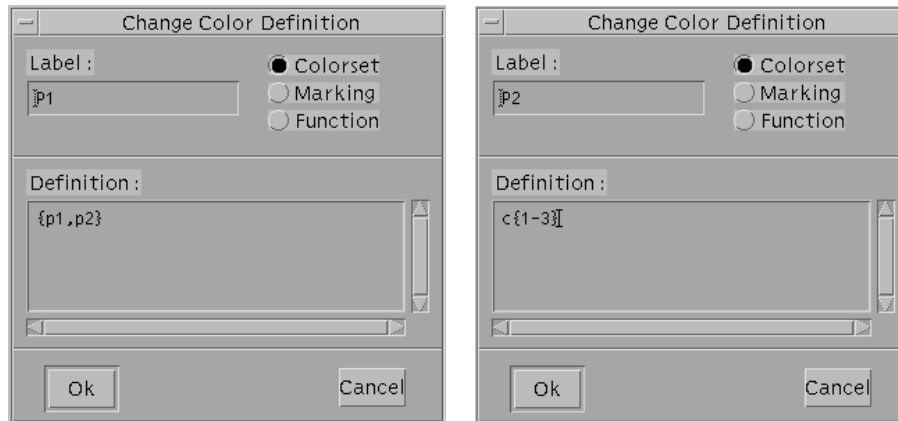


Figure 2.21: Definition of static subclasses.

to the transition *StartW* that is labeled with the *whole place color domain* function $\langle S \rangle$;

- add a guard to the transition *iswrite*. A way to model the constraint that only the processes belonging to the static subclass *P2* are allowed to issue a writing request to the data base is to add a guard to the transition *iswrite*. To modify transition attributes, press one of the *transition* icons and select the interested transition: one of the *Change Transition Properties* windows of Fig.2.15 pops-up, depending on the type of transition, allowing to fill in the “Color Label:” area the guard according to the SWN syntax. In the model of Fig.2.19, transition *iswrite* can fire only when its input place contains a colored token $\langle x \rangle$ belonging to the static subclass *P2*.

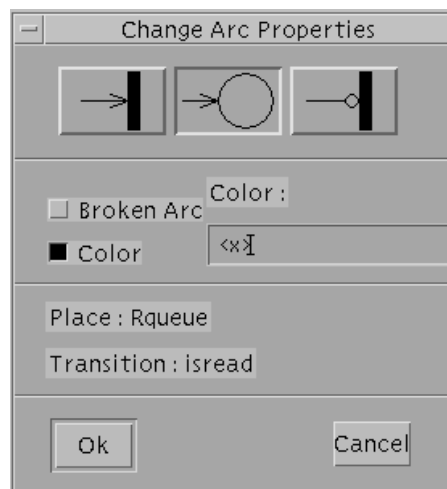


Figure 2.22: Change Arc Properties window.

Finally, we define the colored marking parameter $M0$ of the SWN model of Fig.2.19 by pressing the *color* icon of the *object bar* and by selecting the **Action**→*Add* option. The *Change Color Definition* window pops up again by clicking with the left mouse button on a location in the *canvas*. To define a colored marking parameter switch on the “Marking” toggle and then fill in the “Label:” and the “Definition:” areas with the name of the parameter ($M0$) and its definition ($\langle S\ P1 \rangle + \langle S\ P2 \rangle$) respectively. The initial marking of the SWN model of Fig.2.19, is then set by adding to the “Marking:” area of the *Change Place Properties* window related to the place *think* the colored marking parameter $M0$.

2.7 Analysis of the SWN Readers-Writers model

Concerning the analysis of SWN models, *GreatSPN2.0.2* supports the reachability graph generation (both ordinary and symbolic) with the corresponding Markovian solution, both in steady state and transient, and the simulation: in this section we will describe how to obtain the symbolic reachability graph (SRG) of the Readers-Writers model of Fig.2.19 and its corresponding Markovian solution, for a depth description of the different analysis techniques see Chapt.4.

Once the SWN model has been saved, we can compute the symbolic reachability graph by choosing from the **Swn**→*Symbolic* sub-menu the *Compute RG* option. In case last modifications of the current loaded model have not been saved before launching a *GreatSPN2.0.2* solver a warning window will pop-up asking to the user for saving or aborting the request. The request of computing the symbolic reachability graph of the SWN model will cause the *Console* window of Fig.2.12 to pop-up; it is then possible to obtain a verbose description of the symbolic reachability graph by setting on the “Verbose Show” toggle of the *SWN Symbolic RG Options* window (see Fig.3.19) which appears after the “Start” button of the *Console* window has been pressed.

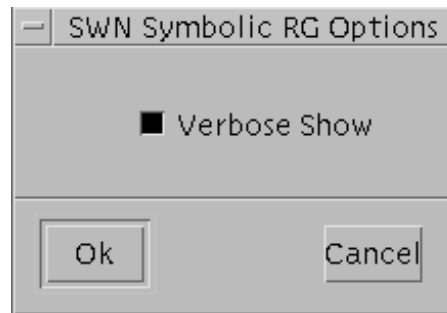


Figure 2.23: SWN Symbolic RG Options window.

Finally, choose “OK” button of the *SWN Symbolic RG Options* window to launch the *GreatSPN2.0.2* solver. The execution is displayed on the *Console* window and the results are visualized in the *GreatSPN2.0.2* canvas: the SWN model of Fig.2.19 is characterized by 45 Tangible Symbolic Markings (which correspond to 209 Tangible Ordinary markings) and no deadlocks are found. Results are saved in different files: the symbolic reachability

graph is saved in file `netname.srgP5`, transition throughputs and performance indices defined by the user are contained instead in `netname.sta` file.