PETRINET SIMULATOR

Program Documentation

Contents

1 Introduction to the Program

The Petrinet Simulator is able to display a petrinet and gradually build a reachability graph by firing transitions. The reachability graph will show whether it encountered a state that is unbounded and mark the beginning and ending nodes on the path signifying the unboundedness of the petrinet. Additionally there is the option to analyse a given petrinet with regards to its boundedness and build the reachability graph with the click of a button. Furthermore it provides an user interface for editing petrinets.

The following sections give an overview of the menus and toolbars and the functionality provided by it.

1.1 The Menu

1.1.1 File

The *File* menu consists of the following entries:

- **New:** opens a new empty file for editing → automatically opens the Editor view (see 1.2.2).
- **Open:** opens saved file from directory.
- **Open in new Tab:** same as open but in a new tab.
- **Reload:** reload the currently opened file.
- **Save:** save changes.
- **Save as:** save changes and choose directory/name.
- **Analyse++:** analyse many petrinets at once. Results are printed in the text area.
- **Close:** close currently opened file.
- **Exit:** close the program.

1.1.2 Edit

The *Edit* menu consists of the following entries:

- **Open Editor:** opens the Editor.
- **Close Editor:** closes the Editor and reverts back to Viewer.
- **Change Look and Feel:** change between Nimbus and Metal look and feel.

1.1.3 Help

The *Help* menu only consists of the element *Info*, which shows information about the Java version used and the user directory.

1.2 Toolbar Modes

The program has two distinct toolbar modes: viewer mode and editor mode. This section discusses the two modes and toolbar buttons associated with it. In general the toolbar is fully de- and reattachable (except for the bottom of the program being reserved for the status bar). All icons for the toolbar buttons are taken from `https://iconoir.com/`.

1.2.1 Viewer Mode

The viewer mode provides functionality for viewing petrinets and building the reachability graph. The toolbar is split into two parts: the petrinet toolbar to the left and the reachability graph toolbar to the right. Below the specifics are discussed.

Petrinet Toolbar 

From left to right the buttons provide the following functionality:

- **Load:** see Menu *File* 1.1.1.
- **Save:** see Menu *File* 1.1.1.
- **Previous:** open the previous file in the current directory → do nothing if current file is the first.
- **Next:** open the next file in the current directory → do nothing if current file is the last.
- **Increment Place:** increment a marked place by one token.
- **Decrement Place:** decrement a marked place by one token.
- **Reset:** reset the petrinet graph to initial state → if place was in-/decremented the initial state is being updated.
- **Zoom in:** zoom in into petrinet graph.
- **Zoom out:** zoom out of patrinet graph.
- **Open Editor:** open the Editor (see 1.2.2).

Reachbility Graph Toolbar 

From left to right the buttons provide the following functionality:

- **Analyse Petrinet:** analyse the currently opened petrinet → the reachability graph is built and a pop up window shows whether the petrinet is bounded or unbounded. If the petrinet is unbounded the starting and ending nodes of the path signifying unboundedness are marked yellow and red in the reachability graph.
- **Reset:** delete the reachability graph except for the initial state (marked gray). Also resets the petrinet.
- **Undo:** undo the last step in the reachability graph. If last step way analysis it is treated as one single step.
- **Redo:** redo the last step in the reachability graph. If last step way analysis it is treated as one single step.
- **Clear text area:** clears the text area.
- **Zoom in:** zoom in into reachability graph.
- **Zoom out:** zoom out of reachability graph.
- **Tree layout:** organizes the reachability graph in a tree layout (see 2.2.1).
- **Circle layout:** organizes the reachability graph in a circle layout (see 2.2.2).
- **Automatic Layout:** uses the auto layout by GraphStream to organize the reachability graph.
- **Reset Split Panes:** resets the split panes to the default ratio.
- **Change Look and Feel:** changes between the different look and feels (*Metal* or *Nimbus*).

1.2.2 Editor Mode

Petrinet Toolbar 

From left to right the buttons provide the following functionality:

- **Load:** see Menu *File* 1.1.1.
- **Save:** see Menu *File* 1.1.1.
- **Previous:** see Petrinet Toolbar in Viewer Mode (1.2.1).
- **Next:** see Petrinet Toolbar in Viewer Mode (1.2.1).
- **Increment Place:** see Petrinet Toolbar in Viewer Mode (1.2.1)..
- **Decrement Place:** see Petrinet Toolbar in Viewer Mode (1.2.1)..
- **Zoom in:** see Petrinet Toolbar in Viewer Mode (1.2.1)..
- **Zoom out:** see Petrinet Toolbar in Viewer Mode (1.2.1)..
- **Add Place:** adds a place above the left top most element.
- **Add Transition:** adds a transition above the left top most element.
- **Remove Element:** removes the marked element (place or transition).
- **Add Edge:** choose beginning and ending node of an edge to add it → beginning and ending nodes have to be of different type (place/transition).
- **Remove Edge:** choose beginning and ending node of an edge to remove it → the edge has to exist.
- **Add Label:** add label to marked place or transition.
- **Close Editor:** closes the Editor and opens the Viewer (see 1.2.1).

Reachability Graph Toolbar 

From left to right the buttons provide the following functionality:

- **Reset Split Panes:** see Reachability Graph Toolbar in Viewer Mode (1.2.1).
- **Change Look and Feel:** see Reachability Graph Toolbar in Viewer Mode (1.2.1).

2 Exploring the Technical Depths

2.1 Design of the Program

Figure 1 shows the general control flow of the program. Double rules signify a GUI component, dashed lines signify interfaces / listeners. The figure only demonstrates the basic design of the program and how the essential data models, controllers and GUI components communicate with each other. Other parts of the program are included in the text as well. The following sections give a detailed description of how the data models are structured and how controllers and listeners work. On a general level it can be noted controllers control the input provided via the GUI elements and listeners transfer changes in the data model to the GUI components.

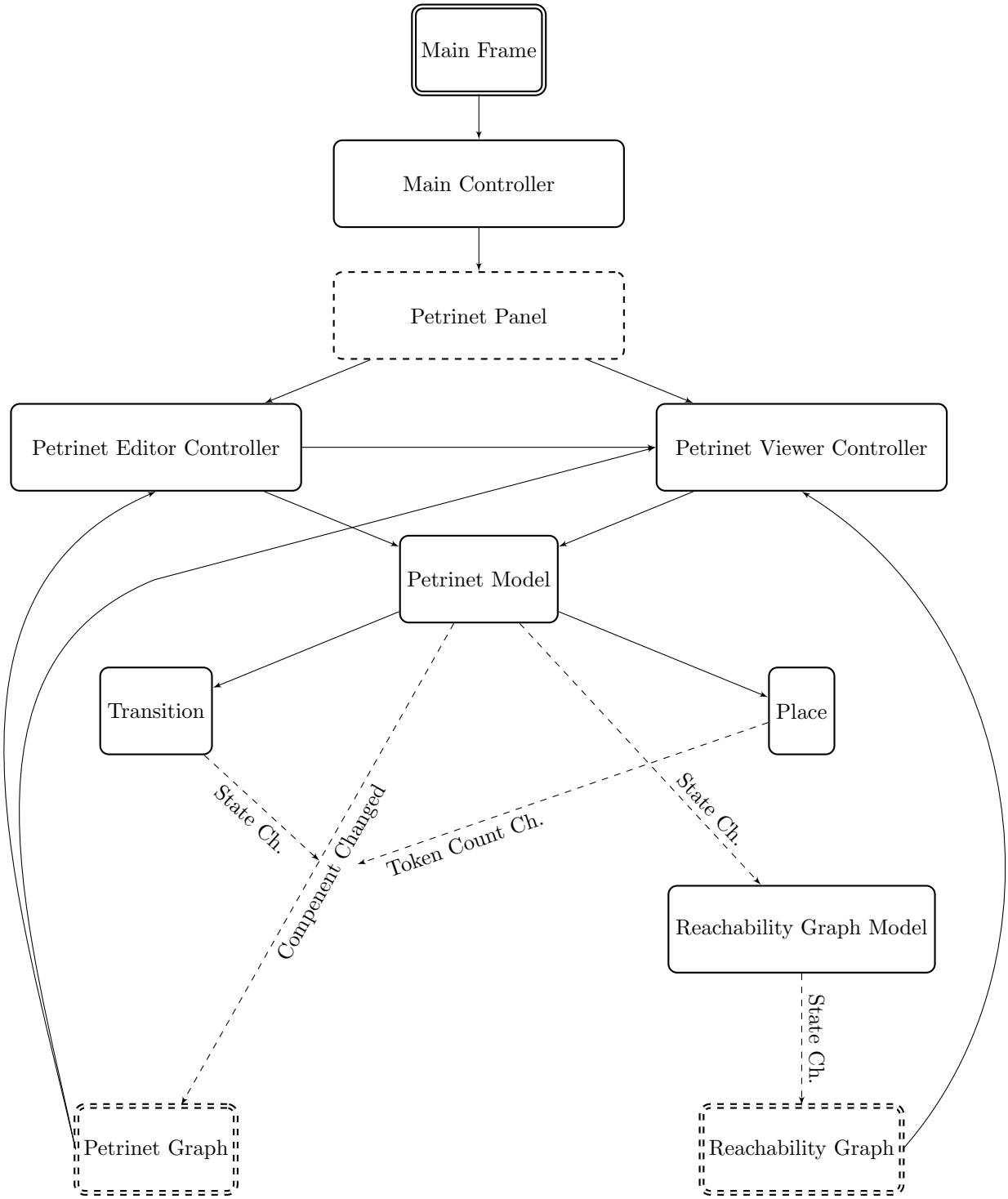

Figure 1: General Design of the Program.

2.1.1 Data Models

Petrinet Model It handles everything belonging only to the petrinet. It has functions for modifying the petrinet (such as adding or removing elements) and checks whether an action is valid. If an action is not valid it throws an according *PetrinetException*. The petrinet model holds helper data structures *Place* and *Transition*. There is no data structure for arcs, as they are fully defined via the places and transitions connected to them[1]. Rather arcs are defined by lists of elements preceding and succeeding places and transitions. Places contain information about the number of tokens stored in them. Transitions have a status of being activated or not activated.

Reachability Graph Model It provides methods for adding/removing states to/from the reachability graph and giving information about the graph. It contains a list of *PetrinetStates*. Such a state contains predecessor and successor states, information about the tokens on the places in the petrinet as well as a state called m, being the first state on a path that marks the given state as unbounded. When a new state is added the model checks whether the new state completes a path signifying unboudedness (see 2.1.4. If so, the state at the start of the path is remembered by the state and the reachability graph model remembers the last state on the path. The states also store transitions leading up to them and from them. These are stored in lists in a map.

Reachability Graph Undo Queue This queue records all the steps taken in the reachabilty graph and keeps track of the actions being taken (i.e. whether something has been added). To this end it has its own data structure called *ReachabilityGraphUndoQueueState*. The queue can go back or forward or can be rewinded/reset. If it goes backward, alls steps are undone. Going forward means redoing all steps. The reachability model has methods to stop pushing steps onto the queue. Furthermore steps can be recorded as "skippable". A step being skippable means on un-/redo it does not stop but continues going back-/forwards. This enables a multi step process like adding steps when analysing the petrinet to be treated as one big step.

2.1.2 Controllers

There are three controllers in the program: the main controller, the petrinet viewer controller and the petrinet editor controller. Additionally the *PetrinetPanelInterface* serves as an intermediary between the main controller and the other controllers.

Main Controller On a general level there is a main frame and a number of petrinets in tabs. The task of the main controller is to pass user interactions with elements pertaining to the main frame (menus and toolbars) to the petrinet panel that is currently active. If tabs are switched it is also the task of the main controller to restore the state of the toolbar with regards to highlighted buttons for the activated petrinet.

Petrinet Panel Interface This interface forwards messages from the main controller. In the implementation of the petrinet panel at hand the panel also handles clicks and forwards them to either the viewer controller or the editor controller.

Petrinet Viewer Controller This controller manages all clicks on the petrinet when in viewer mode and clicks on the reachability graph.

Petrinet Editor Controller This controller manages all clicks on the petrinet when in editor mode. It also is tasked with providing functionality when editing the given petrinet.

2.1.3 Listeners

Petrinet State Changed Listener

[1] For storing ids belonging to arcs a map is used.