CS 6388-01 Model-Integrated Comp | **Petri-Net Design Studio**

Yujie Chi chiy2 yujie.chi.1@vanderbilt.edu

This is a brief introduction to PetriNet Design Studio.

Domain

This studio is developed for Petri-Net model design. Petri-Nets are networks composed of three components: place, transition and arc. The number of tokens in one place is defined as the marking. A transition is said to be enabled if the following two requirements are satisfied: (1) the set of inplaces and the set of outplaces are **both non-empty**; (2) all its inplaces are of at least one token. Fire an enabled transition is to increase one token for each inplace and decrease one token for each outplace. A detailed introduction to concepts regarding Petri-Nets can be accessed at [1].

Applications

Petri-Net is a mathematical modeling language for the description of distributed systems [1]. It is usually applied to model discrete parallel network. An example is website communication: places represent websites and transitions are tunnels or routes. Messages are passed through the network as tokens. Only when one transition is enabled, that is, the transition is connected by a certain number of upstream websites that are holding the information desired by some downstream websites, can this transition be fired. Petri-Nets can also model state machines, marked graphs and workflows.

Installation

You can find the repo through this github link [2]. You should find one folder "modelica-petrinet" which contains all the files you need. Go into this folder.

You should make sure the following installation before getting started:

(1) Nodejs, MongoDB, Git, Python and command line tools of WebGME.

(2) jointjs 2.2.0. (The newest version of jointjs cannot visualize the tokens. This issue is resolved by [3])

(3) The dependencies of jointjs are stored at "./Dependencies".

After clone the repo and get all the necessary packages installed, you can try "node .\app.js" to start the WebGME server. You should be able to find the seed "Modelica" which is the template for users creating their own projects.

You are welcomed to check the code. I only edited these files: "config.default.js", "__init__.py", "MyVisualizerControl.js" and "My VisualizerWidget.js". Every place I edited is with some comments which are easy to find.

[1] https://en.wikipedia.org/wiki/Petri_net
[2] https://github.com/CleverPaul/PetriNetDesignStudio.git
[3] https://github.com/clientIO/joint/issues/860

Usage

The visualizer we are using is “MyVisualizer”. You should double check if the visualizers and plugins you need (“MyVisualizer”,”ModelEditor”,”ModelicaCodeGenerator”) are selected.

There are several examples you can play with:

(1) “example”.

(2) “freechoice”, a model that can be categorized as free-choice Petri-Net.

(3) “statemachine”, a model that can be categorized as state machine.

(4) “markedgraph”, a model that can be categorized as marked graph Petri-Net.

(5) “WorkflowNet”, a model that can be categorized as workflow net.

(6) “noarcs”, a model composed of one place, two transitions and no arcs.

You are welcomed to create your own Petri-Nets. You can refer to the following steps:

(1) Go to ROOT, drag a “PetriNet” block from bottom left.

(2) Go into this PetriNet, create places and transitions. Change their names and set the markings of all places.

(3) Connect places and transitions.

(4) Click “MyVisualizer” on the left to visualize the model and do simulations.

The visualizer/simulator has the following functions:

(1) The model you created is visualized in the manner called bipartite. One column is for places and one column is for transitions.

(2) Places are visualized along with their tokens. Place name is in the format “name-marking”.

(3) Enabled transitions are in color green while unenabled are in red.

(4) Click one enabled transition to fire it. The tokens are drawn as small black disks when the amount is small or as a number when large.

(5) Click the up-arrow button in the toolbar to go to its parent (upper level).

(6) Click the left-arrow button in the toolbar to reset the model to its initial state.

(7) Click the right-arrow button to check the categorization of the model, an alert window will show up.

(8) When the model reaches deadlock (either initially it’s deadlock or after several steps of firing), an alert window will pop up.

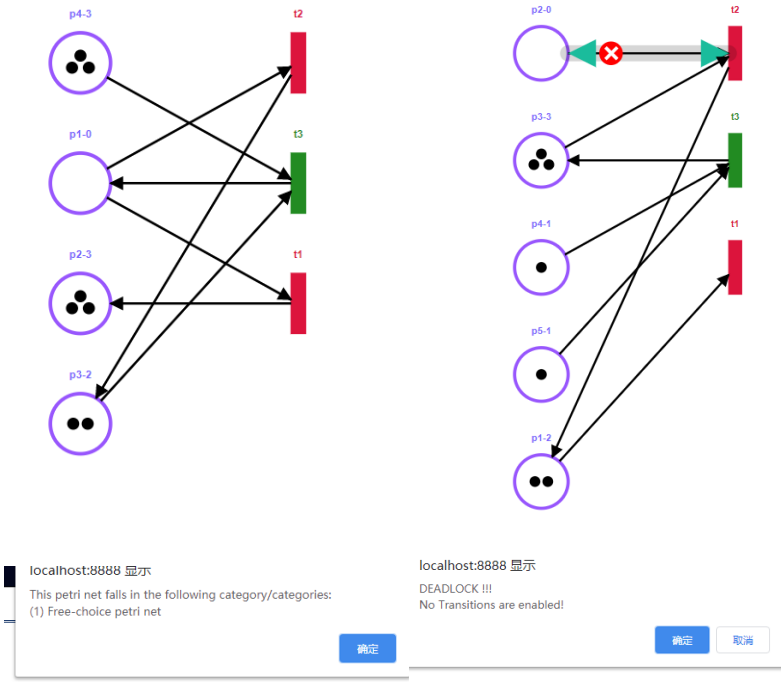
[1] https://en.wikipedia.org/wiki/Petri_net
[2] https://github.com/CleverPaul/PetriNetDesignStudio.git
[3] https://github.com/clientIO/joint/issues/860

Notes:

You should note that:

(1) It may take several seconds to initially visualize the model because the JS function setTimeOut() is used to guarantee messages are received.

(2) Here are some examples:



[1] https://en.wikipedia.org/wiki/Petri_net
[2] https://github.com/CleverPaul/PetriNetDesignStudio.git
[3] https://github.com/clientIO/joint/issues/860