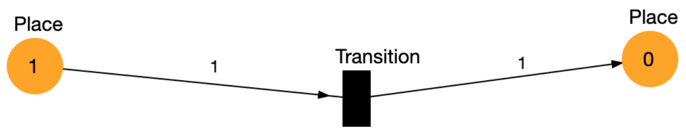
Petri Net Design Studio

Diego Manzanas Lopez, Ayana Wild, Carlos Olea

Petri Net Overview

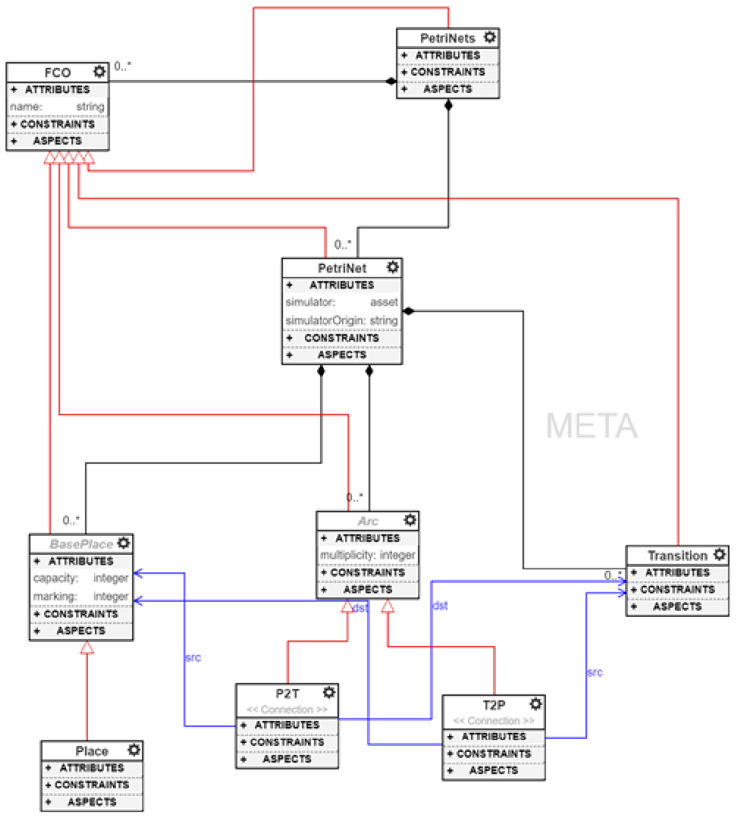
Petri Nets are graphical and modeling tools used to describe and study information processes systems of various types. Petri Nets were first introduced in 1962 by Carl Adam Petri as the topic of his dissertation. They are used to describe and reason about the flow of information in concurrent systems. Petri Nets are fundamentally composed of places, transitions, arcs, and tokens. Places may or may not have a specified capacity as well as a number of tokens currently held at that place (a marking). Transitions are entities that control token flow from place to place. When a transition is fired, a number of tokens is transferred from the place(s) prior to the transition, and from the transition to the subsequent place(s). Arcs are the connections between transitions and places. Since Petri Nets are bipartite, weighted directed graphs, arcs may only point from place to transition or transition to place and have a multiplicity (weight) dictating the number of tokens to be moved. While Petri Nets may contain many more features, the aforementioned properties comprise a low-level Petri Net. An example of a Petri Net from our visualizer is shown below.



Metamodel and Examples

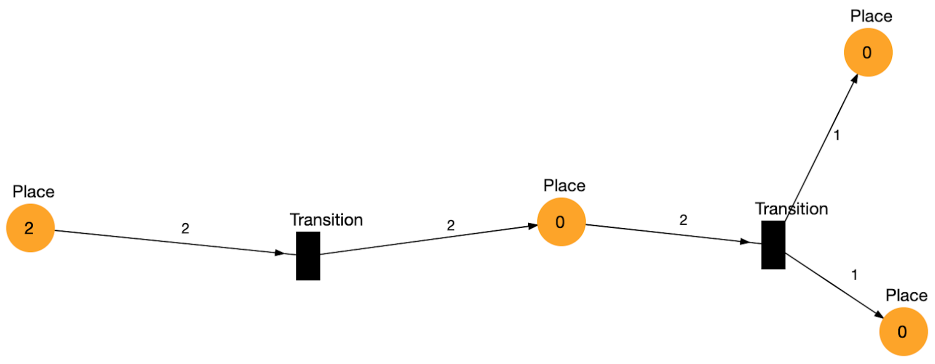
In this project, we created a Design Studio for composing and simulating general Petri Nets. Our WebGME implementation provides a simulator (*PetriNetSim*) and JavaScript plugin (*PetriNetCodeGenerator*) to simulate and analyze Petri Nets. Our metamodel, shown below, describes the basic components of a Petri Net. Through utilizing the components from the metamodel, we are able to create and specify places, transitions, arcs between places and transitions, and arcs between transitions and places. We can also define the multiplicity of arcs, number of tokens in places (marking), and the capacity of the places. We provide a code generator plugin to simulate the composed Petri Net. Once the code generator plugin is executed, a button to initialize the simulator and an interactive event executor appears. Our implementation fires one transition, specified by the user, at a time instead of firing all of the transitions simultaneously. This follows the Petri Net firing rule:

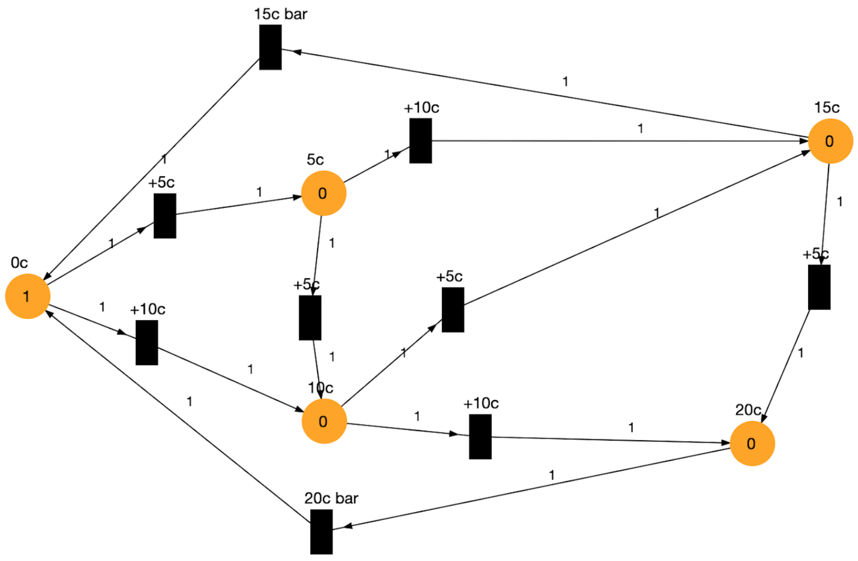
- A transition cannot be fired until the number of tokens at the places prior to said transition is greater than or equal the sum of the multiplicities of the incoming arcs.

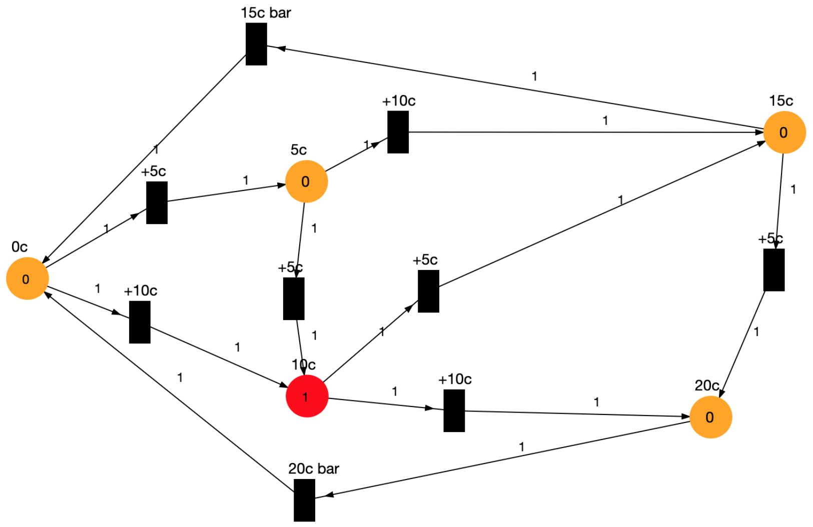


Our simulator allows the user to interact with the Petri Net and see how the tokens move throughout the Petri Net. If the user tries to move more tokens than the place contains, the simulator asks them to restart the model and follow the instructions in the prompt.

Below are some example Petri Nets from our simulator. The one on the top is a modification of the simple example above. The one in the middle is an example simulating a vending machine. Also, the one on the bottom is an example of the same vending machine mid simulation. For all of the examples, you can change the multiplicities of the arcs, the markings of the places, the names of the places, and the overall layout of the Petri Net. All of the edits must be done in the Composition view. After any edits, the *PetriNetCodeGenerator* must be executed to update the data model that is used by the simulator.


Place
2
2
Transition
2
Place
0
2
Transition
1
Place
0
1
Place
0


15c bar
15c
+10c
5c
+5c
0c
+10c
+5c
+5c
10c
+10c
20c
20c bar
+5c


15c bar
15c
+10c
5c
+5c
0c
+10c
+5c
+5c
10c
+10c
20c
20c bar
+5c

Installation and execution process:

1. Ensure you have WebGME properly installed.
2. Clone the following PetriNets repository: https://github.com/dieman95/Petri-Nets
3. In the repository directory run ‘webgme start’
4. Open a browser and go to localhost:8888
5. Create a new project using the seed **PetriNet.webgmex** found in src/seeds/PetriNet
6. Execute the **PetriNetCodeGenerator** plugin and refresh the webpage
7. Choose an example or build a custom PetriNet
8. Press the **Initialize** button when looking at the **PetriNetSim** panel
9. Follow the prompts in the textbox at the top of the simulator
10. Enter an amount and press **Go!**
11. Repeat as many times as desired or until a terminal state is reached.