Documentation

Christian Meter, *Heinrich-Heine-University Düsseldorf*

This project is about developing a petri net in the function language Clojure. In this document the decisions for the design of the project are described. The main object was to stay simple and functional.

1 Architecture Decisions

For this project a generic basic approach was chosen to encapsulate the program as follows:

- The *logic* of the program is implemented in `core.clj`. Without checking for invalid input all commands are executed on the DSL.
- On top of the logic there is the *API* in `api.clj`. Here are the functions which can be used by the simulator and the GUI.
- The *Simulator* uses functions from the API to make (random) modification of the petri net possible.
- The *GUI* visualizes the current DSL.

1.1 Controller

Logic All functions are here to initialize and manipulate the petri nets are in the Controller.

No inspection for valid input To keep the functions simple, none of them will check any input. They only do what has been typed into the REPL. So the liability totally depends on the user's input. That is okay, because later on will be publi functions available to manipulate the DSL including an inspection.

Simplicity Nearly every function was built simple. Only `merge-net` was a bit more difficult, because it calls all the little merge-functions and combines every result to get the new merged net. But everything inside this function is not really complected but because of it's size it is hard to say if it is still simple.

Manipulating State Only few functions are needed to manipulate the DSL. Because there must be a place where the database of petri nets is modified, the functions are reduced to a minimum.

1.2 API

If possible return value, else nil In the API are functions built in which check the user's input and executes it if possible. If not, it will always return `nil`. This is the idiomatic way to handle false inputs by the user and this is why it was chosen.

Built on top of the controller Every function needed to directly manipulate the DSL, is provided by the API. So the user does not need to get into the pure logic of this program to work with the petri nets.

1.3 Simulator

JavaFX vs. SeeSaw

1.4 GUI

JavaFX vs. SeeSaw