Dokumentation

Christian Meter, *Heinrich-Heine-Universität Düsseldorf*

Hier werden die Architekturentscheidungen meines Petrinetz erläutert. Außerdem gehe ich darauf ein, wie Simplicity erreicht wurde, wo Tradeoffs auftraten und wo komplexe Konstrukte nötig waren.

1 Architekturentscheidungen

Ich wähle einen generischen Ansatz und kapsele mein Programm wie folgt: Die komplette Logik befindet sich im *Core*. Dort wird jede Operation ohne Überprüfung durchgeführt.
Darauf baut die *API* auf. Die API stellt alle Funktionen zur Manipulation der Datenbank zur Verfügung.
Dazu gibt es einen *Simulator* und eine GUI die nur Funktionen benutzen, die von der API überprüft und zur Verfügung gestellt wurden.

1.1 core.clj

Logik In dem Core werden alle Funktionen implementiert, die zur Manipulation der Netze notwendig sind.

Keine Kontrolle auf valide Eingaben Um die Funktionen möglichst simple zu halten, wird in core.clj bzw. der Ausführung in der REPL darauf verzichtet zu überprüfen, ob die Ausführung einer Funktion möglich ist. Die Überprüfung erscheint mir an dieser Stelle nicht sinnvoll und möchte die Verantwortung dafür an den Benutzer selbst abgeben.

Das bedeutet *nicht*, dass gar keine Überprüfung stattfinden wird. Die GUI wird später auf Funktionen zurückgreifen, die vorher sicherstellt, dass die gewünschte Aktion möglich ist.

1.2 API api.clj

If possible return value, else nil Die API wird bei einer validen Eingabe den entsprechenden angefragten Wert zurückliefern. Andernfalls wird *nil* zurückgegeben.

Bereitstellung der Daten aus dem Core Jede Funktion, die für die Manipulation der Netze benötigt wird, wird in der API zur Verfügung gestellt. Hilfsfunktionen im Core natürlich nicht.

1.3 Simulator - simulator.clj

JavaFX vs. SeeSaw

1.4 GUI - view.clj

JavaFX vs. SeeSaw