Praktikumsaufgaben 2 : Spezifikation in Renew

Version vom: 24. März 2014

Aaufgabe 2.1 Die Fibonacci-Zahlen sind folgendermaßen definiert:

$$f(n+2) = f(n+1) + f(n), \quad f(1) = 1, \quad f(0) = 0$$

1. Entwerfen Sie ein gefärbtes Petrinetz, das eine Stelle *fibs* besitzt und das die folgende Schaltfolge erlaubt
$$\mathbf{m}_0 \xrightarrow{*} \mathbf{m}_1 \xrightarrow{*} \cdots \xrightarrow{*} \mathbf{m}_n \xrightarrow{*} \cdots ,$$
wobei die Markierung der Stelle *fibs* in den $\mathbf{m}_n, n \in \mathbb{N}$ die folgende Multimenge ist:
$$\mathbf{m}_n(\mathit{fibs}) = 1'(0, f(0)) + \cdots 1'(n-1, f(n-1)) = \sum_{i=0}^{n-1} 1'(i, f(i))$$
Für die Markierung evtl. vorhandener weiterer Stellen wird nichts gefordert.
Entwerfen Sie das Netz so, dass die Übergäng von $\mathbf{m}_i \xrightarrow{*} \mathbf{m}_{i+1}$ durch das Schalten genau einer Transition (in möglicherweise verschieden Bindungen) realisiert wird.
2. Geben Sie alle Komponenten Ihres Netzes explizit an!
3. Implementieren Sie ihr Netz in *Renew*, simulieren sie es bis $\mathbf{m}_4$ und drucken Sie den Simulationsstatus aus!
Im Ausdruck sollen die Marken sichtbar sein, nicht nur die Markierungskardinalitäten. Zu erreichen mittels folgender Aktionen: (1) Edit → Select All → Nodes → Places und (2) Net → Marking → Tokens.
4. Variieren Sie das gefärbte Netz derart, dass die Markierung der Stelle *fibs* die $(0, f(0)), \ldots, (k-1, f(k-1))$ nicht als Multimenge, sondern als Sequenz $\langle f(k-1), \ldots, f(0)\rangle$ enthält. (Diese Reihenfolge ist am einfachsten zu erzeugen.) Verwenden Sie dazu den *Renew*-Listenoperator $\{\mathit{head} : \mathit{tail}\}$.

Aaufgabe 2.2 Ein Fahrstuhl fährt zwischen den Stockwerken 1 und n. Er kann ein Stockwerk nach oben fahren (Aktion: up), ein Stockwerk nach unten fahren (Aktion: down), auf einem Stockwerk i halten (Aktion: stop) und die Fahrtrichtung umkehren (Aktion: turn)

In jedem Stockwerk y kann ein Fahrstuhl angefordert werden (Aktionen: req(i) mit $1 \leq i \leq n$).

In der Kabine kann ein Zielstock ausgewählt werden. (Aktionen: req(i) mit $1 \leq i \leq n$).

Die Steuerungslogik des Fahrstuhls ist die folgende: Er speichert alle Halteanforderungen, die für Stockwerke oberhalb seines aktuellen gelten, in einer Menge *ReqAbove*. Getrennt davon die Anforderungen für Stockwerke unterhalb *ReqBelow*.

Fährt die Kabine nach oben, so fährt sie weiter nach oben, solange es noch Anforderungen oberhalb gibt. Anderfalls wird die Richtung geändert. Analog, falls die Kabine nach nach unten fährt. Initial steht die Kabine im Stockwerk 1, Fahrtrichtung ist nach oben. Gibt es für das aktuelle Stockwerk eine Anforderung, so wird gestoppt.

1. Modellieren Sie dieses Szenario als gefärbtes Petrinetz. Verwenden Sie den in der Vorlesung präsentierten Editor RENEW, um das Netz zu erstellen!

2. Testen Sie das Modell, indem Sie im Simulationsmodus von RENEW zufällige Halteanforderungen generieren.

In Ihrem Laborbericht geben Sie auch Mitschnitte Ihrer Session an.