Softwarekonstruktion – Übung 3

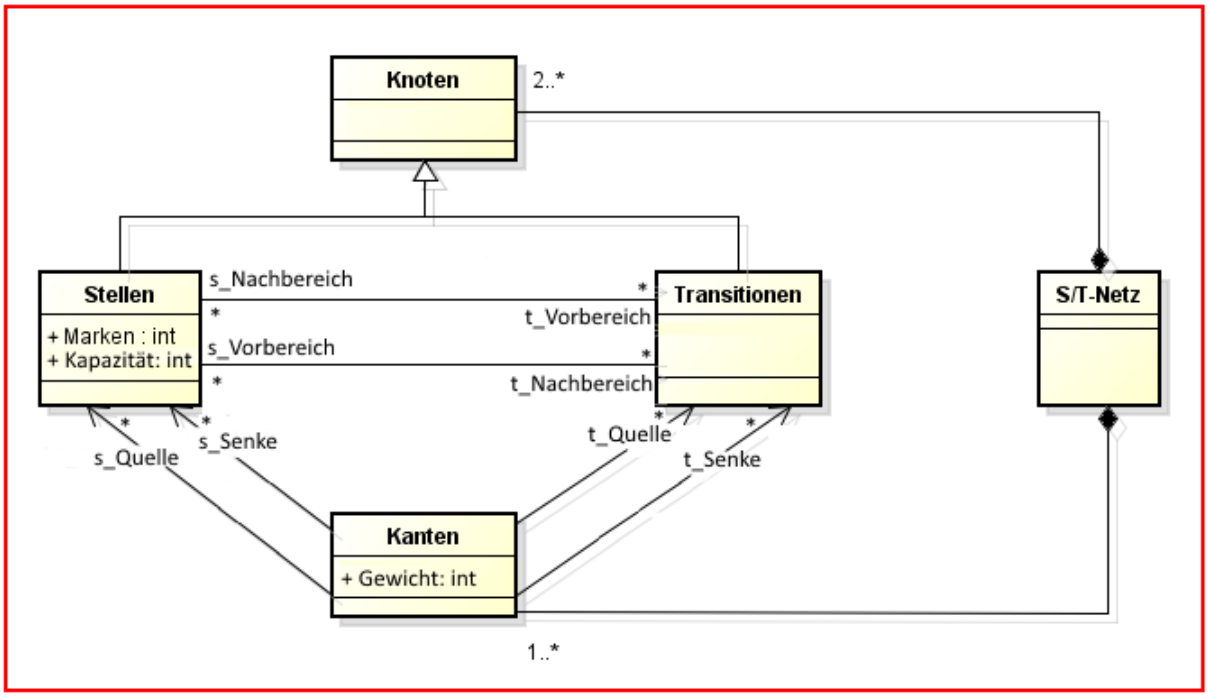
3 Petrinetze + Metamodellierung

3.1 Petrinetze und Metamodelle

Gegeben sei folgende Definition eines S/T-Netzes:
Ein S/T-Netz $\mathsf{ST} = (\mathsf{S}, \mathsf{T}, \mathsf{F}, \mathsf{K}, \mathsf{W}, \mathsf{M}_0)$ ist ein 6-Tupel mit:

- $(\mathsf{S}, \mathsf{T}, \mathsf{F})$ ist ein Petri-Netz, d.h.
  - S ist endliche, nicht-leere Menge von Stellen
  - T ist endliche, nicht-leere Menge von Transitionen
  - S und T sind disjunkt (d.h. $S \cap T = \emptyset$) und bilden die sog. *Knotenmenge* des S/T-Netzes
  - F ist endliche, nicht-leere Menge von Flüssen (Kanten) mit $F \subseteq (S \times T) \cup (T \times S)$
- $\mathsf{K} : \mathsf{S} \to \mathbb{N} \cup \{\infty\}$ ist Kapazitätsfunktion, die jeder Stelle eine Kapazität zuordnet. **Achtung: Bei der Bearbeitung der dieser und der folgenden Aufgabe sei zur Vereinfachung die default-Kapazität ∞ nicht zulässig!**
- $\mathsf{W} : \mathsf{F} \to \mathbb{N}_+ \setminus \{0\}$ ist Gewichtsfunktion, die jeder Kante ein positives Gewicht zuordnet.
- $\mathsf{M}_0 : \mathsf{S} \to \mathbb{N}$ ist eine Markierungsfunktion (Anfangsmarkierung), die jeder Stelle eine Anzahl von Marken zuordnet, wobei gilt $\forall \mathsf{s} \in \mathsf{S} : \mathsf{M}(\mathsf{s}) \leq \mathsf{K}(\mathsf{s})$.

Geben Sie ein Metamodell in UML für die Modellierung von S/T-Netzen an.
Modellieren Sie dabei folgenden Begriffe: S/T-Netz, Knoten, Stellen, Marken, Kapazität einer Stelle, Transitionen, Kanten, Vorbereich, Nachbereich, Quelle einer Kante, Senke einer Kante, Gewicht einer Kante (Kantenvielfachheit) **Achtung: in dieser und der Aufgabe 3.2 sei zur Vereinfachung die default-Kapazität ∞ nicht zulässig!**. **Verwenden Sie die 5 Klassen Knoten, Transitionen, Stellen, Kanten und S/T-Netz! Achten Sie beim Modellieren darauf, dass alle Rollennamen eindeutig sind!**



3.2 Metamodelle und OCL

Untersuchen Sie Ihr in Aufgabe 3.1 erstelltes Metamodell und stellen Sie fest, ob sich damit unzulässige Petrinetze modellieren lassen. Falls ja, an welcher Stelle?
Benutzen Sie OCL-Ausdrücke in dem Kontext *Kante*, um die Modellierungsschwäche zu beheben!

```
context Kanten
(1) inv: ((self.t_Quelle -> union (self.s_Quelle))->size()=1) and
         ((self.t_Senke -> union (self.s_Senke))->size()=1)
Aussage: Jede Kante hat genau 1 Quelle und genau 1 Senke

(2) inv: (self.s_Quelle -> notEmpty() implies self.s_Senke -> isEmpty()) and
         (self.t_Quelle -> notEmpty() implies self.t_Senke -> isEmpty())
Aussage: Bipartiter Graph, so dass Senke und Quelle jeder Kante unterschiedlichen
Typs sind

Alternative für (1) und (2):
    inv: (self.t_Quelle->size()=1 and self.s_Senke->size()=1 and
          self.t_Senke->isEmpty() and self.s_Quelle->isEmpty()) xor
         (self.t_Senke->size()=1 and self.s_Quelle->size()=1 and
          self.t_Quelle->isEmpty() and self.s_Senke->isEmpty())
```

```
(3) inv: self.s_Quelle -> notEmpty() implies
         (self.s_Quelle.t_Nachbereich -> includes (self.t_Senke) and
          self.t_Senke.s_Vorbereich -> includes (self.s_Quelle))
    inv: self.t_Quelle-> notEmpty() implies
         (self.t_Quelle.s_Nachbereich -> includes (self.s_Senke) and
          self.s_Senke.t_Vorbereich -> includes (self.t_Quelle))
Aussage: Stellen und Transitionen besitzen Vor- und Nachbereich. Zu einer gegebenen
Kante müssen Quelle und Senke in diese Beziehung zueinander gebracht werden.
Dafür werden Senken einer Kante in die Menge des Nachbereichs der Quelle und
Quellen einer Kante in die Menge des Vorbereichs der Senke aufgenommen. Dies
muss sowohl für Stellen als auch für Transitionen als Quellen gelten.
Anmerkung: Notation hier erlaubt, da durch (1) gesichert wurde, dass die Mengen
(z.B. self.t_Senke) nur 1 Element enthalten; 1-elementige Mengen dürfen als
Objekt angesprochen werden

(4) inv: (self.s_Quelle -> union (self.s_Senke))-> forAll(s:Stelle |
         s.Marken >= 0 and s.Kapazität > 0 and s.Marken <= s.Kapazität)
Aussage: Für alle Stellen (egal ob Quelle oder Senke einer Kante) muss gelten,
dass sie keine negativen Marken enthält, die Kapazität größer als 0 ist und
dass die Anzahl der Marken der Stelle die Kapazität der Stelle nicht übersteigt

(5) inv: self.Gewicht > 0
Aussage: Das Kantengewicht ist immer positiv
```

3.3 Petrinetze - Netzeigenschaften

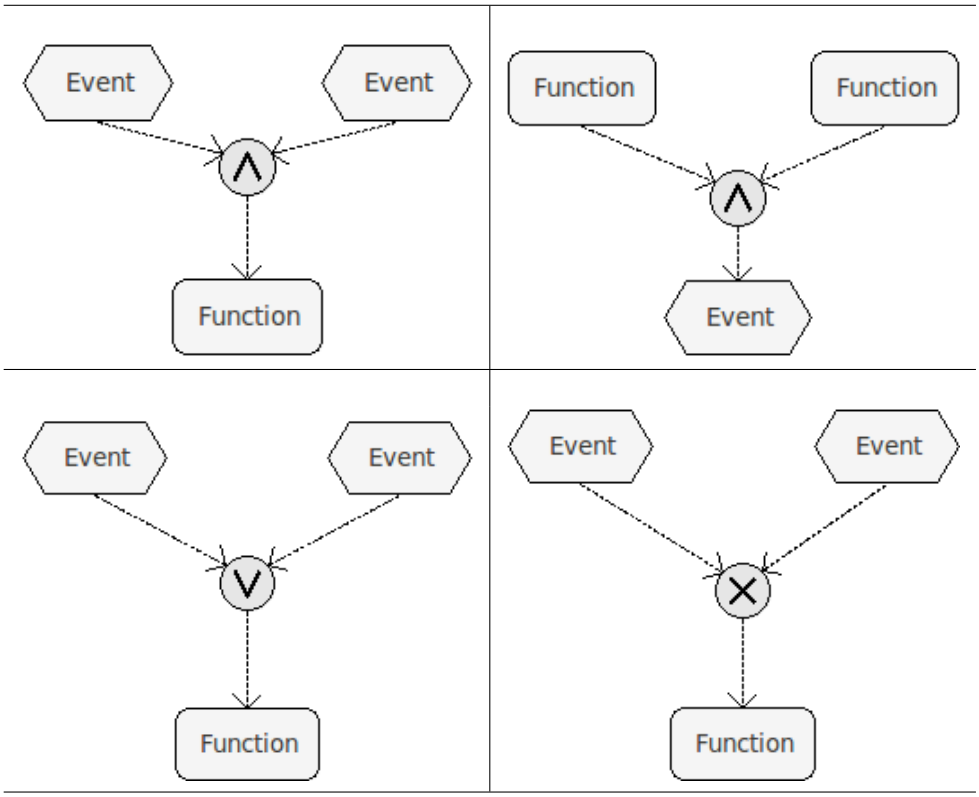
Sei $ST = (S, T, F, W, K, M_0)$ ein S/T-Netz.

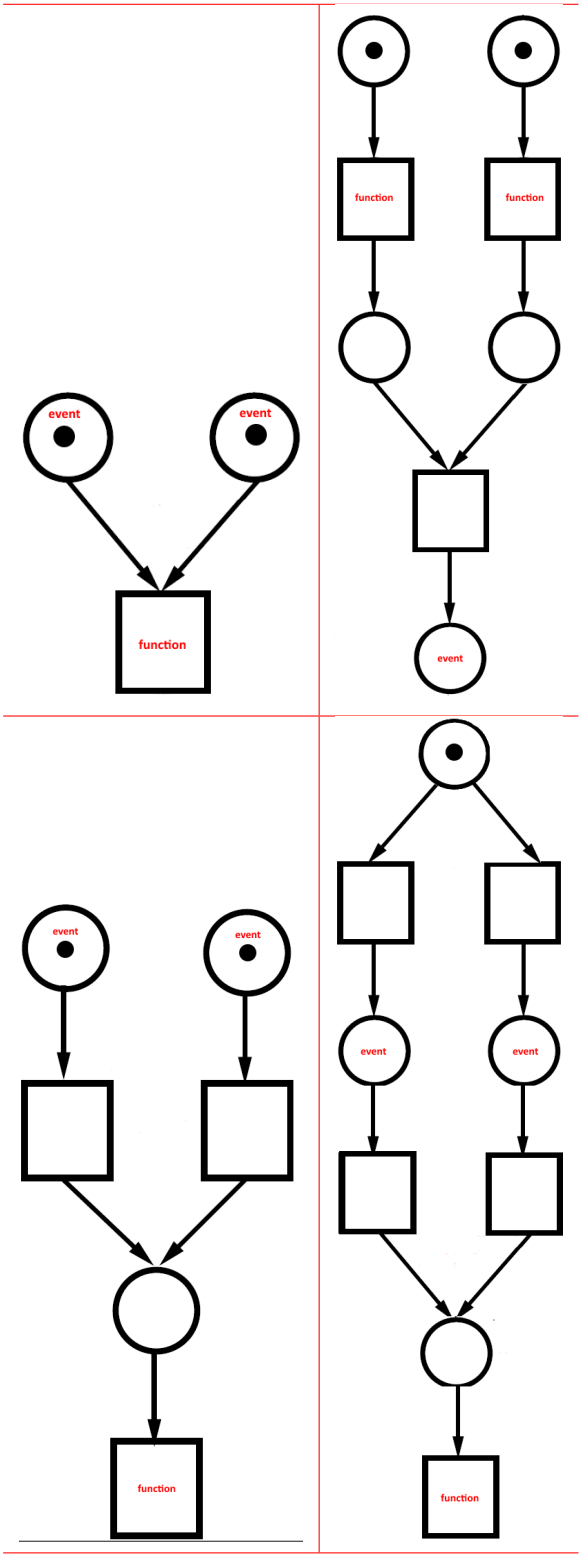
1. Erklären Sie die formalen Definitionen aus den Vorlesungsfolien umgangsprachlich und verständlich.
   - Eine Transition $t \in T$ heißt **aktivierbar im gesamten Netz** $\Leftrightarrow$ $\exists M_i \in [M_0 >$ mit: $M_i[t >$
   - Eine Transition $t \in T$ heißt **lebendig im gesamten Netz** $\Leftrightarrow$ $\forall M_i \in [M_0 >$ gilt: $\exists M_j \in [M_i >$ mit: $M_j[t >$
   - Eine Transition $t \in T$ heißt **tot im gesamten Netz** $\Leftrightarrow$ $\forall M_i \in [M_0 >$ gilt: $\neg M_i[t >$
2. Erklären Sie den Zusammenhang zwischen einer lebendigen Tansition und einem lebendigen S/T-Netz sowie zwischen einer toten Transition und einem toten S/T-Netz umgangsprachlich.

1. formale Definitionen umgangsprachlich erklären:
   - Eine Transition $t \in T$ heißt im ganzen Netz aktivierbar, genau dann wenn **mindestens eine** von der Startmarkierung M_0 aus erreichbare Markierung M_1 existiert, in der t aktiviert ist, d.h. schalten könnte.
   - Eine Transition $t \in T$ heißt im ganzen Netz lebendig, genau dann wenn für **alle** von der Startmarkierung M_0 aus erreichbare Markierungen M_i jeweils **mindestens eine** von M_i aus erreichbare Folgemarkierung M_j existiert, in der t aktiviert ist, d.h. schalten könnte.
   - Eine Transition $t \in T$ heißt im ganzen Netz tot, genau dann wenn für **alle** von der Startmarkierung M_0 aus erreichbare Markierung M_i gilt, dass t in M_i **nicht** aktiviert ist, d.h. **nicht** schalten könnte. Dies ist äquivalent zu der Aussage, dass **keine** von M_0 aus erreichbare Markierung M_i existiert, in der t aktiviert ist, d.h. schalten könnte.
2. Bei einem lebendigen Netz müssen alle Tansitionen in der Startmarkierung lebendig sein. In einem toten Netz müssen alle Transitionen in der Startmarkierung tot sein.

3.4 Petrinetze - Transformation

Transformieren Sie folgende EPK-Ausschnitte in gültige Petrinetze.


Event
Event
Function
Function
Function
Event
Event
Event
Function
Event
Event
Function


event
event
function
function
function
event
event
event
function
event
event
function