Nested Petri Nets

Seminar Logik und Semantik von Programmen (FGI 3)
Wintersemester 2011/2012,

Alexander Beifuß, Robert Keßler, Tim Krämer
{7beifuss,8kessler,7kraemer}@informatik.uni-hamburg.de

Universität Hamburg

Zusammenfassung *Nested Petri nets* (zu deutsch: "Netze in Netzen") Stellen einen Ansatz dar der darauf ausgelegt ist verteilte Systeme in Hinblick auf Hierarchie, Nebenläufigkeit, Mobilität und Kapselung zu modellieren.

Keywords: Nested Petri nets, Erweiterte Workflow-Netze, Entscheidbarkeit, Netze in Netze, Synchronisation, Simulation und Verifikation verteilter Systeme

Inhaltsverzeichnis

1 Motivation

Gerade in Komplexen Systemen kommt es häufig zu Schwierigkeiten bei der Modellierung real weltlicher Sachverhalte. Dies ist besonders bei hierarchischen, verschachtelten Strukturen der Fall. Dafür eigenen sich jedoch Nested Petri Nets sehr gut, da mit ihnen eine formale Aufteilung in Subprozesse möglich ist, wobei sich jedoch Prozess und Subprozess in ihrer Notation - jeweils einzeln betrachtet - kaum unterscheiden. Somit sinkt beispielsweise der Aufwand für eine Verifikation des Gesamtsystems, da weniger Verfahren aufgrund der Ähnlichkeit der Systeme entwickelt werden müssen.

Einsetzen lässt sich das Ganze beispielsweise in der Medizin bei der Modellierung von Prozessabläufen [2]. Hier können allgemeine medizinische Vorgehensweisen als Prozesse und die jeweilige Durchführung der komplexeren Analysen und Behandlungen als Subprozesse modelliert werden.

2 Grundlagen

Dieser Abschnitt soll dem Leser zur Auffrischung, des in dieser Arbeit behandelten Themas, dienen. Es werden sowohl grundlegende Definitionen und Konzepte bereit gestellt sowie auch die verwendete Notation festgelegt und eine Terminologie bereit gestellt. Die Informationen (insbesondere die Definitionen) auf denen dieser Abschnitt basiert stammen aus [1].

2.1 Netze

Definition 2.11 *Seien P und T zwei disjunkte Mengen und $F \subseteq (P \times T) \cup (T \times P)$. Dann bezeichnet das Tupel $\mathcal{N} = (P, T, F)$ ein Netz, wobei die Elemente der jeweiligen Mengen wie folgt bezeichnet werden:*

$p \in P$ Platz oder Stelle
$t \in T$ Transition
$f \in F$ Fluss-Relation (gerichtete Kanten, Kanal)

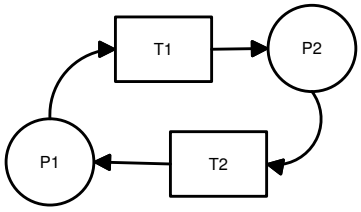

Abbildung 1. Beispielgrafik eines einfachen Netzes

Definition 2.12 *Sei $x \in P \cup T$, dann ist der Vorbereich (eines Platzes oder einer Transition) definiert durch ${}^\bullet x := \{y \mid (y,x) \in F\}$ und der Nachbereich durch $x^\bullet := \{y \mid (x,y) \in F\}$.*

Als nächstes führen wir die Mengen I und O für die eingehenden und ausgehenden Kanäle (in dieser Reihenfolge) einer Transition ein, welche später bei der Beschriftung der Kanäle benötigt werden.

Definition 2.13 *Für eine Transition t, ergibt sich die Menge der Eingangs-Kanäle zu $I = \{(x,t) \mid x \in {}^\bullet t\} \subseteq F$*
und die der Ausgangs-Kanäle zu $O = \{(t,x) \mid x \in t^\bullet\} \subseteq F$.

2.2 Markierung

Eine Markierung M beschreibt den Zustand eines Netzes und wird durch eine Funktion beschrieben, bei der jede Stelle $p \in P$ auf die auf ihm befindlichen Marken abgebildet wird. Bei „einfachen“ Petri Netzen ist die Markierung eine Funktion $m : P \to N$, bei der jeder Platz des Netzes auf die Anzahl, der auf ihm befindlichen Marken (Es gibt nur *„black dot tokens“* - *„schwarze Marken“*), abgebildet wird. Da es jedoch wünschenswert ist gleich von Anfang an beliebig komplexe Markierungen zulassen, führen wir das Konzept von Multimengen ein.

Definition 2.21 *Eine Multimenge m über einer Grundmenge S ist eine Abbildung von S nach $\mathbb{N}$ ($m : S \to \mathbb{N}$), wobei $\mathbb{N}$ die Menge der natürlichen Zahlen ist. $m(s)$ gibt dabei an wie oft $s \in S$ in m vorkommt.*

Weiterhin soll S_{MS} die Menge aller Multimengen über S bezeichnen. Diese wird auch oft als $\mathbb{N}_0^S$ notiert und kann ähnlich einer Potenzmenge interpretiert werden, wobei die Potenzmenge über einer Menge gebildet wird in der jedes Element aus S abzählbar unendlich oft vorkommt. Nach diesem Prinzip erhält man eine Menge, in der jede Multimenge, die über S gebildet werden kann, enthalten ist.

Beispiel:
$S = \{\heartsuit, \clubsuit\}$
$S_{MS} = \{\emptyset, \{\heartsuit\}, \{\clubsuit\}, \{\heartsuit, \clubsuit\}, \{\heartsuit, \heartsuit\}, \{\clubsuit, \clubsuit\}, \dots\}$

Definition 2.22 *Eine Multimenge m über S bezeichnet man als endlich gdw. die reduzierte Grundmenge $\{s \in S \mid m(s) > 0\}$ der Multimenge endlich ist. Die reduzierte Grundmenge (auch als $supp(m)$notiert) beschreibt die Menge der 'relevanten' Elemente einer Multimenge m und entspricht damit der Grundmenge S.*

Definition 2.23 *Seien m und m' Multimengen, dann beschreiben wir mit $m \leq m'$, die Inklusion von m in m'. m ist in m' enthalten gdw. $m(x) \leq m'(x) \forall x \in supp(m) \cup supp(m')$*

Definition 2.24 *Seien m und m' Multimengen, dann beschreibt $m + m'$ die Addition zweier Multimengen mit $m + m' = \{m(x) + m'(x) \mid x \in supp(m) \cup supp(m')\}$*

Beispiele:
$\{\heartsuit, \heartsuit, \clubsuit\} < \{\heartsuit, \heartsuit, \clubsuit, \clubsuit\}$
$\{\heartsuit, \clubsuit, \heartsuit, \clubsuit\} = \{\heartsuit, \heartsuit, \clubsuit, \clubsuit\}$
$\{\heartsuit, \clubsuit\} + \{\clubsuit, \heartsuit\} = \{\heartsuit, \heartsuit, \clubsuit, \clubsuit\}$

Es sei darauf hingewiesen, dass durch eine einelementige Grundmenge S, Markierungen mit 'schwarzen Marken' weiterhin möglich sind. Hierbei handelt es sich um einen Spezialfall. Allgemein lassen sich jedoch nahezu beliebig komplexe Objekte (mit einer endlichen Struktur) darstellen, da Multimengen über beliebigen Grundmengen beliebig geschachtelt werden können.

Definition 2.25 *Sei nun $\mathcal{N} = (P, T, F)$ ein Netz und S eine beliebige Grundmenge, dann nennt man eine Funktion $M : P \to S_{MS}$, welche jede Stelle $p \in P$ auf eine Multimenge über S abbildet, eine Markierung $\mathcal{N}$ über S.*

Bei einer gegebene Markierung M und einem Netz $\mathcal{N} = (P, T, F)$ spricht man auch von einem markierten Netz. Wobei M als initiale Markierung bezeichnet wird.

Beispiel:
$S = \{\heartsuit, \clubsuit, \dots\}$
$M(p_1) = \{\heartsuit, \clubsuit, \clubsuit\}$
$m_{p_1}(\heartsuit) = 1$
$m_{p_1}(\clubsuit) = 2$

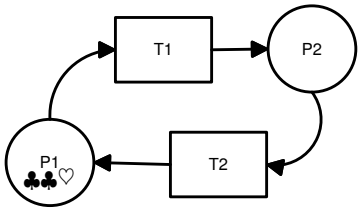

Abbildung 2. Beispielgrafik eines gefärbten Petri-Netzes

2.3 Kanalbeschriftung

Als nächstes wird das Konzept der Kanalbeschriftung eingeführt, welches benötigt wird, wenn man Petri Netz Modelle verwenden will, in denen das Schalten

von Transitionen dazu führen kann, dass Marken entweder ersetzt werden oder aber, dass zwei (oder mehr) Marken nach dem Schalten auf unterschiedlichen Plätzen des Nachbereiches der Transition liegen. Um die Ausdrücke, mit denen die Kanäle beschriftet werden können, zu definieren, führen wir zuerst zwei endliche, disjunkte Mengen $V = \{v_1, \dots\}$ und $C = \{c_1, \dots\}$ ein. V sei die Menge der Bezeichner für Variablen und C sei die Menge der Bezeichner für Konstanten. Dann ist $B = V \cup C$ die Menge aller terminalen Bezeichner (oft auch als „atome“ bezeichnet).

Definition 2.31 *Ein Ausdruck e ist eine endliche Multimenge über die atomaren Bezeichner, $e : B \to \mathbb{N}$.*

Ausdrücke werden üblicherweise nicht in Mengenform notiert, sondern meist in der Form einer Summation, da dieses etwas intuitiver zu lesen ist.

Beispiel:
$\{\heartsuit, \clubsuit, \clubsuit, \spadesuit\} = \heartsuit + \clubsuit + \clubsuit + \spadesuit$

Die Menge aller möglichen Ausdrücke über einer Menge B von Bezeichnern ist gegeben durch $B_{MS} = \mathbb{N}_0^B$ (weiterhin als $Expr(B)$ notiert). Sie ergibt sich aus allen möglichen Multimengen über der Grundmenge der Bezeichner B (siehe oben).

$Var(e)$ sei eine Funktion welche die Menge der Variablenbezeichner liefert, die in einem Ausdruck $e \in Expr(B)$ vorkommenden.

Angenommen eine Konstante c bezeichne ein festes Element c_s der Menge S und b sei eine Funktion, welche eine Variable v auf ein Element $b(v) \in S$ abbildet (und für Elemente aus C deren Identität beibehält), dann bezeichnet $b(e)$ eine Multimenge über S. Denn beim Anwenden von b auf eine Multimenge $e \in Expr(B)$ wird b nur Einfluss auf Elemente von V nehmen.

2.4 Transitionsbeschriftung

Als nächstes wird das Konzept der Transitionsbeschriftungen näher betrachtet, diese ermöglichen später das synchrone feuern (oder auch schalten) zwei Transitionen. Zunächst werden zwei disjunkte Mengen von Etiketten benötigt. Aus diesem Grund werden $Lab = \{l_1, l_2, \dots\}$ und $Lab' = \{l^1, l^2, \dots\}$ mit $Lab \cap Lab' = \emptyset$ eingeführt.

Weiterhin seinen $Lab = \{l_1, l_2, \dots\}$ und $Lab' = \{l^1, l^2, \dots\}$ zwei disjunkte Mengen von Kanalbeschriftungen. Für jede Kanalbeschriftung $l \in Lab \cup Lab'$ wird nun eine benachbarte Kanalbeschriftung $\bar{l}$ definiert, so dass die Mengen $Lab, Lab', \overline{Lab} =_{def} \{\bar{l} \mid l \in Lab\}$ und $\overline{Lab'} =_{def} \{\overline{l'} \mid l' \in Lab\}$ paarweise disjunkt sind. Dieses bedeutet, dass die Schnittmenge zweier beliebiger Mengen $\emptyset$ entsprechen muss.
Weiterhin sei $\bar{\bar{l}} =_{def} l$ und $\mathcal{L} =_{def} Lab \cup Lab' \cup \overline{Lab} \cup \overline{Lab'}$

2.5 Netze-in-Netzen

Definition 2.51 *Eine Netze-in-Netzen-Struktur Σ ist eine Anordnung von $i \geq 1$ Netzen, wobei $\mathcal{N}_1$ als System-Netz und alle anderen als Element-Netz(e) bezeichnet werden.*
Jedem eingehenden (bzw. ausgehenden) Kanal innerhalb eines Netzes $\mathcal{N}_i = (P_i, T_i, F_i)$ wird nun mit einem Ausdruck $\mathcal{E}(p,t)$ (bzw. $\mathcal{E}(t,p)$) aus $Expr(B)$ beschriftet (der Ausdruck darf auch leer sein). Wobei die Forderung besteht, dass weder Bezeichner von Konstanten in den Kanalbeschriftungen eines beliebigen Eingangs-Kanals auftauchen, noch dass Bezeichner von Variablen doppelt bei den Ausdrücken an den Eingangs-Kanälen einer Transition vorkommen dürfen (siehe Abbildung 3).

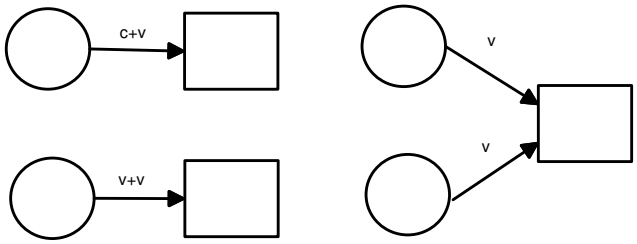

Abbildung 3. Beispiele „verbotener“ Beschriftungen für eingehende Kanäle an Transitionen (nachempfunden aus [1])

Diese Restriktionen sind notwendig, damit bestimmte Eigenschaften im Netz erhalten bleiben. (dazu später mehr).

Gegeben sei eine Nested-Petri-Net-Struktur Σ, wobei die Markierung der Element-Netze $\mathcal{N}_2, \ldots, \mathcal{N}_i$ über die endlichen Mengen $S_2, \ldots S_i$ interpretiert werden und $\mathcal{M}$ die Menge aller markierten Element-Netze aus Σ darstellt.

Definition 2.52 *Ein Nested-Petri-Net (NPN) ist eine Netsted-Petri-Net-Struktur Σ, bei der jede Konstante $c \in C$ als eine Menge markierter Element-Netz aus $\mathcal{M}$ betrachtet wird.*

Wenn von der Markierung eines NPN gesprochen wird, so bezieht sich dieses auf die Markierung des System-Netzes über der Menge $\mathcal{M}$.

Die Definition eines NPN hängt damit ebenfalls von den Mengen $S_2, \ldots, S_i$ ab. Handelt es sich bei den S_i um (Grund-)Mengen mit nur einem Element, so sind die Element-Netze einfache Petri-Netze mit einfarbigen Marken. Verwendet man hingegen mehr-elementige (Grund-)Mengen, so handelt es sich bei den Element-Netzen um gefärbte Petri-Netze. Es ist aber auch möglich, dass die Element-Netze zugleich System-Netze sind, indem ihre Markierung Mengen von markierten Netzen als Marken erlaubt. So erhält man Neted-Petri-Net-Strukturen mit beliebig vielen Ebenen. Auch können diese Mengen wiederum

das System-Netz $\mathcal{N}_1$ als Elemente enthalten, wodurch man schließlich sogar zur Rekursion gelangt.

2.6 Schaltverhalten

Das Schalten von Transitionen in Neted Petri Nets erfordert das Instanzieren von Variablen in den Etiketten der Übergänge.

Definition 2.61 *Sei $\mathcal{N}_i = (P_1, T_i, F_i)$ ein Netz innerhalb eines Neted Peti Net. Die Bindung einer Transition $t \in T_i$ geschieht mittels der Funktion b, welche jede Variable $v \in V$ auf einen Wert $b(v) \in \mathcal{M} \cup S_2 \cup S_3 \cup \cdots \cup S_i$ abbildet.*
Eine gebundene Transition ist dabei ein Paar $Y = (t, b)$ mit $t \in T$ und der Bindung b von t.
Eine gebundene Transition $Y = (t, b)$ ist schaltbar in einer Markierung M von $\mathcal{N}_i$ gdw. $\forall p \in {}^\bullet t : b(\mathcal{E}(p, t)) \subseteq M(p)$.
Schaltet eine gebundene Transition Y in der Markierung M, so überführt sie das Netz in die Markierung M' (Notation: $M \xrightarrow{Y} M'$). Für jedes $p \in P_i$ folgt $M'(p) =_{def} M(p) - b(\mathcal{E}(p, t)) + b(\mathcal{E}(t, p))$.
Markierte Element-Netze, welche als Variablenwert innerhalb eines Ausdrucks $\mathcal{E}$ eines eingehenden Übergangs dienen, werden mit „am Schalten von t beteiligt“ bezeichnet.

Definition 2.62 *Gegeben sei ein NPN. Das Schalten des NPN lässt sich durch vier unterschiedliche Typen charakterisieren.*

Transport-Schritt: *Eine nicht etikettierte Transition schaltet in System-Netz. Dabei bleiben die Element-Netzte unbeeinflusst. Dieser Typ wird Transport-Schritt genannt, da lediglich Element-Netze (ggf. auch schwarze Marken) im System-Netz bewegt werden.*

Objekt-autonomer-Schritt: *Schaltet eine nicht etikettierte Transition innerhalb eines Element-Netz, so bleiben das System-Netz und alle anderen Netze unbeeinflusst. Es ändert sich nur der Interne Zustand des schaltenden Element-Netzes.*

horizontale Synchronisation: *Das gleichzeitige Schalten von zwei Transitionen aus zwei Element-Netzen, welche auf dem selben Platz liegen (unter Beachtung der gleichen Bindung) wird durch die Etiketten l und $\overline{l}$ aus $Lab' \cup \overline{Lab'}$ unterstützt. Es wird von horizontaler Synchronisation gesprochen, da sich beide Netze (aus hierarchischer Sicht) auf der selben Ebene befinden.*

vertikale Synchronisation: *Hiermit wird das gleichzeitiges Schalten, einer mit $l \in Lab \cup \overline{Lab}$ etikettierten Transition t des System-Netzes und einer mit $\overline{l}$ etikettieren Transition eines Element-Netzes, welche am Schalten von t beteiligt ist, bezeichnet.*

3 Beispiel eines NPN

Anhand eines kleinen Beispiels soll nun das Konzept von Netze-in-Netzen erläutert werden. Als Beispiel soll ein NPN dienen (siehe Abbildung **??**), welches das Scheduling (sehr abstrakt) von Threads/Prozessen modelliert. Das System-Netz soll die Zuweisung der CPU an einen Thread/Prozess (Element-Netz(e)) übernehmen. Um die horizontale Synchronisation zu verdeutlichen, sollen die Threads in der Lage sein sich zu synchronisieren — wobei die Synchronisation sehr abstrakt ist und im Weiteren nicht darauf eingegangen wird, welche genaue Bedeutung diese hat.

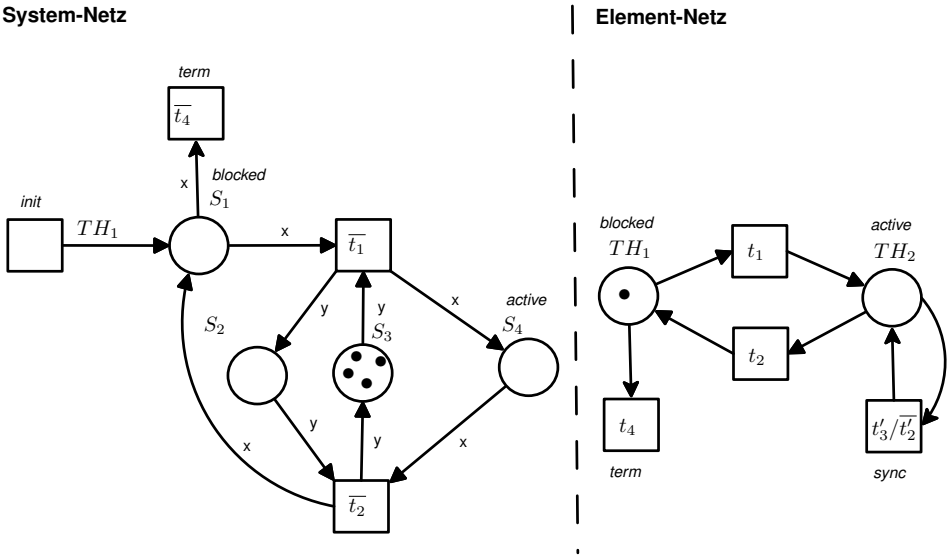

Abbildung 4. Beispiel für ein NPN. System-Netz (links), Element-Netz (rechts)

Die Element-Netze (kurz: EN) besteht aus vier Transitionen und zwei Stellen. Folgen werden nun die Bedeutungen der Stellen erläutert.

Befindet sich eine schwarze Marke auf der Stelle TH_1, so befindet sich der „Thread“ (bzw. das Element-Netz) in dem Zustand auf dem er auf die Zuweisung der CPU wartet. Der Thread gilt in diesem Fall als „rechen-willig“. Liegt hingegen die Marke auf der Stelle TH_2, so repräsentiert dieses einen „Thread“, der gerade aktiv ist („ausgeführt“).
Das System-Netz (kurz: SN) besteht hingegen aus vier Transitionen und vier Stellen.

Die nicht etikettierte Transition von SN (mit der Notiz „*init*“) dient der Generierung eines neuen Threads, welches durch ein Element-Netz (kurz: EN) mit

der Markierung TH_1 dargestellt wird (bei der Kantenbeschriftung TH_1 in NS handelt es sich um eine Konstante — Es sei weiterhin angemerkt, dass diese Transition die einzige ist, welche anfangs schalten kann). Feuert diese Transition, so wird daher ein ein Thread mit der Markierung TH_1 (EN) der Markierung von S_1 (SN) hinzugefügt. Element-Netze, welche Markierungen der Stelle S_1 sind, sind daher „rechen-willige Threads“ und warten auf die Zuweisung der CPU. Element-Netze, welche zu der Markierung von S_4 gehören, stellen Threads dar, welche gerade „ausgeführt“ werden. Die Stellen S_2 und S_3 heben sich etwas von den anderen beiden Stellen ab, da ihre Markierungen nicht aus Element-Netze bestehen sondern aus schwarzen Marken. Die Anzahl der Marken auf S_2 gibt die Anzahl der gerade „ausgeführten“ Threads wieder und die Marken der Stelle S_3 gibt die Anzahl der noch verfügbaren Kerne wieder (in dem Beispiel gibt es insgesamt vier schwarze Marken. Daher)

Die unetikettiere Transition zur Generierung der Threads wurde schon erklärt, nun sollen die anderen Transitionen (beziehungsweise synchronisierten Transitionspaare) erläutert werden. Die synchronisierten Transitionen $\overline{t_1}$ und t_1 dienen dem Dispatching des Threads, bei dem die Zuweisung der CPU an einen Thread stattfindet. Wird dieses Transitionspaar geschaltet, so wird im SN ein durch die Kantenbeschriftung an die Variable „x“ gebundener Thread und eine an die Variable „y“ gebundene schwarze Marke auf die Stellen S_4 beziehungsweise S_2 verschoben. Im Element-Netz hingegen, sorgt die synchronisierte Transition t_1 für den Übergang des Threads in den Zustand, in dem der Thread als „ausgeführt“ angesehen wird. Für aktive Threads gibt es nun verschiedene Möglichkeiten des Schaltens. Wenn beispielsweise ein zweiter Thread gerade ausgeführt wird, so kann das Transitionspaar $\overline{t_3'}$ und t_3' feuern und eine Synchronisation der beteiligten Threads nachstellen. Wird hingegen t_2 geschaltet, so geht der Thread wieder in den blockierten Modus zurück und wird durch $\overline{t_2}$ im System-Netz ebenfalls in den Zustand verschoben, dessen Markierung blockierte Threads sind (S_1). Das Feuern von $\overline{t_2}$ hat ebenfalls den Effekt, dass eine Marke, welche die zugewiesene CPU symbolisiert, wieder nach S_3 verschoben wird und somit wieder als „verfügbar“ gilt. Nun kann der Threads gegebenenfalls wieder aktiviert werden durch $\overline{t_1}$ und t_1 oder aber durch $\overline{t_4}$ und t_4 terminieren. Für den Fall, dass der Thread terminieren soll, wird die Marke im Element-Netz entfernt und das Element-Netz selbst aus dem System-Netz entfernt (somit ist dieses Element-Netz nicht mehr vom System-Netz aus zugänglich). Es gibt sicherlich viel ausführlichere Modellierungen für die Zuweisung der CPU an einen Thread, jedoch soll dieses auch nur ein überschaubares Beispiel sein, welches die Semantik der möglichen Schaltdynamiken erläutert.

4 Petri Nets mit rücksetzenden Kanten

An dieser Stelle möchten wir auf den Vergleich zwischen Nested Petri Nets und Petri Nets mit rücksetzenden Kanten (reset arcs) hinweisen. Diese erweitern das

Grundmodell um die Möglichkeit, durch Ausführung bestimmter Transitionen, ensprechende Stellen zu leeren (reset).

Satz 4.01 *Petri Nets mit rücksetzenden Kanten können durch Nested Petri Nets mit gewöhnlichen Petri Nets als Objektnetze simuliert werden.*

Beweis: Die Idee ist das Vorhandensein von n Marken auf einer Stelle durch ein einfaches Elementnetz mit n Marken zu simulieren. Dann ist es möglich den *reset*-Effekt einer rücksetzenden Kante zu simulieren, indem man das Elementnetz mit $E0$ ersetzt, einem neuem Elementnetz mit Null Marken. Das Inkrementieren und Dekrementieren von Marken auf einer Stelle, kann durch äquivalente Operationen im entsprechenden Elementnetz simuliert werden, denn diese werden durch den Synchronisationsprozess erzwungen.

5 Entscheidbarkeit

Kritische Probleme bei der Verifikation von Petri Netzen sind z.B. das *Halte Problem*, *Erreichbarkeit* und auch *Begrenztheit*. Dieser Abschnitt soll nun einen kleinen Überblick über die gängigsten Probleme und ihre Entscheidbarkeit geben.
Halte Problem: Ein Netz terminiert, falls es nicht unendlich ausgeführt werden kann.
Erreichbarkeit: Eine Markierung M' ist von M aus erreichbar, wenn eine Schaltfolge existiert mit der man von M zu M' gelangt.
Begrenztheit: Die Erreichbarkeits-Menge eines Netzes ist die Menge aller Markierungen, welche aus der Initial-Markierung (Start-Markierung) erreichbar sind.
Instandhaltbarkeit: Gegeben ist eine Initial-Markierung M sowie eine endliche Menge $Q = q_1, q_2, \ldots, q_m$ an Markierungen. Es gilt zu entscheiden ob eine Schaltfolge existiert, bei der jede Markierung der Zwischenschritte einem der q_i entspricht.
Unvermeidbarkeit: Es gilt zu entscheiden ob alle Schaltfolgen startend bei gegebener Initial-Markierung M eine Markierung besuchen, welche keinem der q_i entspricht.
Dieser Abschnitt soll einen kleinen Überblick über die Entscheidbarkeit der kritischen Probleme bei der Verfifikation

Satz 5.02 Erreichbarkeit *und* Begrenztheit *sind für Netsted Petri Nets unentscheidbar.*

Beweis: Nach [1] sind *Erreichbarkeit* und *Begrenztheit* für Petri Netze mit rücksetzenden Kanten unentscheidbar. Da wie oben gezeigt Nested Petri Nets auch Petri Netze mit rücksetzenden Kanten simulieren können folgt direkt, dass unentscheidbare Probleme für Petri Netze mit rücksetzenden Kanten auch für Netsted Petri Nets unentscheidbar sind.
Somit ergibt sich direkt, dass sowohl *Erreichbarkeit* als auch *Begrenztheit* für Nested Petri Nets unentscheidbar sind.

Um eine Antwort auf die Frage zur Entscheidbarkeit zu erhalten, nutzen wir das Konzept der wohl geformten Transitionssystemen.
Zur Wiederholung: Ein Transitionssystem ist ein Tupel $\mathcal{S} = \langle S, \rightarrow \rangle$ indem $\mathcal{S}$ ein abstraktes Set an Zuständen (oder Konfigurationen) darstellt und $\rightarrow \subseteq \mathcal{S} \times \mathcal{S}$ für jede Transitionsrelation steht. Für ein Transitionssystem $\mathcal{S} = \langle S, \rightarrow \rangle$ schreiben wir $Succ(s)$ für das Set $\{s' \in \mathcal{S} | s' \rightarrow s\}$ von direkten Nachfolgern von s. $\mathcal{S}$ ist endlich verzweigt, wenn alle $Succ(s)$ endlich sind.

Ein wohlgeformtes Transitionssystem ist ein Transitionssystem mit einer kompatiblen Quasi-Wohlordnung.
Zur Wiederholung: Eine Quasi-Ordnung ist jede reflexive und transitive Relation $\leq$ (über ein Set X).

Satz 5.03 *Quasi-Wohlordnung (well-quasi-ordering) beschreibt jede Quasi-Ordnung $\leq$, sodass für jede endliche Sequenz $x_0, x_1, x_2, ...$ in X Indizes $i \leq j$ existieren, sodass $x_i \leq x_j$ gilt.*

Satz 5.04 *Ein wohlgeformtes Transitionssystem ist ein Transitionssystem $\Sigma = \mathcal{S}\langle S, \rightarrow, \leq \rangle$ mit einer Ordnung $\leq \subseteq \mathcal{S} \times \mathcal{S}$ zwischen Zuständen, sodass $\leq$ eine Quasi-Wohlordnung und $\leq$ 'kompatibel' zu $\rightarrow$ ist, wobei 'kompatibel' meint, dass für alle $s_1 \leq t_1$ und Transitionen $s_1 \rightarrow s_2$ eine Transition $t_1 \rightarrow t_2$ existiert, sodass $s_2 \leq t_2$ gilt.*

Sei NPN ein Nested Petri Netz, $\mathcal{M}_{MS}$ das Set aller Zustände.

Eine Quasi-Ordnung $\leq$ auf dem Set $\mathcal{M}_{MS}$ ist folgendermaßen definiert:
für $M_1, M_2 \in \mathcal{M}_{MS} : M_1 \leq M_2$, wenn für alle $p \in P_{\mathcal{N}_1}$ gilt, dass eine injektive Funktion $\jmath_p : M_1(p) \rightarrow M_{2(p)}$ existiert, sodass $\forall \langle \mathcal{N}, m \rangle \in M_1(p)$, für $s \in M_1(p)$ gilt entweder $\jmath_p(s) = s$ oder $s = \langle \mathcal{N}_i, m \rangle$ und $\jmath_p(\langle \mathcal{N}_j, m \rangle) = \langle \mathcal{N}_j, m' \rangle$ für $m \leq m'$.

oder in kürzer: $M_1, M_2 \in \mathcal{M}_{MS} : M_1 \leq M_2$, wenn für alle $p \in P_{\mathcal{N}_1}$ gilt, dass eine injektive Funktion $\jmath_p : M_1(p) \rightarrow M_{2(p)}$ existiert, sodass für $\forall \langle \mathcal{N}, m \rangle \in M_1(p)$ gilt $\jmath_p(\langle \mathcal{N}_i, m \rangle) = \langle \mathcal{N}_i, m \rangle \vee \langle \mathcal{N}_i, m' \rangle$ mit $m \leq m'$.

Abbildung 5 zeigt ein Beispiel für zwei Markierungen M_1, M_2 eines NP-Netzes, geordnet nach $\leq$. Das abgebildete System hat drei Plätze p_1, p_2, p_3. Der einzige Typ eines Elementnetzes hat die Plätze q_1, q_2. In beiden Markierungen ist Platz p_1 leer und Platz p_2 enthält einen Netztoken, aber die Markierung des Netztokens in M_1 ist in der entsprechenden Markierung M_2 enthalten. Der Platz p_3 in M_2 enthält den selben Netztoken, wie in M_1 plus einen Netztoken mehr. Dementsprechend ist die Relation $\leq$ ein Art Nested Set Inklusion.

These:: Sei NPN ein Nested Petri Netz, mit $\mathcal{M}_{MS}$ als Set für alle Zustände des NPN, $\rightarrow$ die Flussrelation auf $\mathcal{M}_{MS}$, und $\leq$ die Quasi-Ordnung auf $\mathcal{M}_{MS}$, wie oben definiert. dann ist $\langle \mathcal{M}_{MS}, \rightarrow, \leq \rangle$ ein wohl geformtes Transitionssystem.

Beweis. $\leq$ ist eindeutig eine Quasi-Wohlordnung, aber wir müssen noch zeigen, dass sie kompatibel zu zur Transitionsrelation $\rightarrow$ ist. Dafür haben wir folgende Fallunterscheidung:

1. Sei $M_1 \rightarrow^t M_1'$ ein gültiger Schritt über eine Transition t und sei $M_1 \leq M_2$. Dann existiert für jede transferierte Marke $s \in M_1(p)$ ein Objekt $\jmath_p(s) \in M_2(p)$. Wegen der Eingabebeschränkungen werden alle Objekte unabhängig

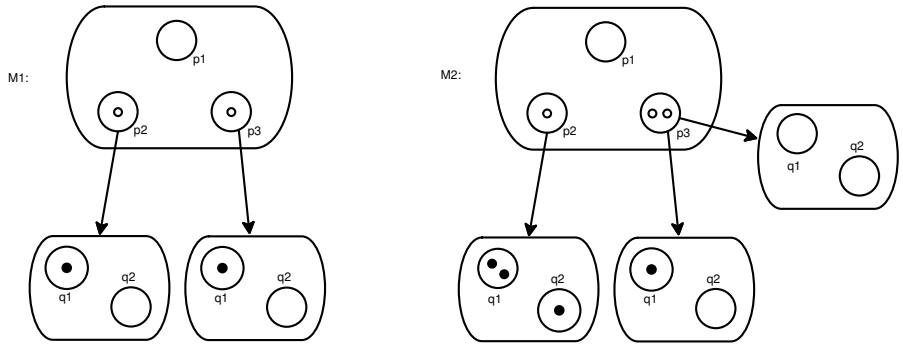

Abbildung 5. Beispiele für Markierungen $M_1 \leq M_2$ (nachempfunden aus [1])

transferiert und das Schalten von t hängt nicht von Markierungen ab. Es ist einfach zu erkennen, dass wenn $M_2 \rightarrow^t M_2'$, dann $M_1' \leq M_2'$.

2. Für einen objectautonomen Schritt ist die gegebene Kompatibilität offensichtlich.
3. Eine horizontale Synchronisation ist die gleichzeitige Ausführung diverser objectautonome Schritte. Die Kompatibilität kann analog zum vorherigen Schritt gezeigt werden.
4. Ein vertikaler Synchronisationsschritt ist die gleichzeitige Ausführung eines Transportschritts und diversen objectautonomen Schritten. Die Kompatibilität ergibt sich zwar nicht direkt aus den ersten beiden Fällen, kann aber durch Kombination der vorherigen Beweise bewiesen werden. □

Man beachte, dass wir kein wohl geformtes Transitionssystem hätten, falls wir ein mehrfaches Vorkommen von Variablen an einer Eingangskantenbeschreibung erlauben würden. Daraus folgt, dass Object Petri Netze von Valk et. al. keine wohlgeformten Transitionssysteme sind.

Es wurde in [1] bewiesen, dass

- das Halteproblem ist entscheidbar für wohlgeformte Transitionssysteme mit (1) transitiver Kompatibilität, (2) entscheidbarer Quasi-Ordnung $\leq$, und (3) gültigen $Succ(s)$.
- Instandhaltbarkeit und Unvermeidbarkeit sind entscheidbar für wohl geformte Transitionssysteme mit (1) Übergangskompatibilität, (2) entscheidbarer Quasi-Ordnung $\leq$, und (3) gültigen $Succ(s)$.

Es folgt, dass für Netze in Netzen folgende Lemma gelten:

Lemma 1 *1. Die Quasi-Ordnung $\leq$ ist entscheidbar.*
2. Succ ist ein gültiger Nachfolger.

Theorem 1. *Das Halteproblem ist entscheidbar für Netze in Netzen. (Nested Petri Nets)*

Der Beweis folgt aus der Annahme zur Quasi-Ordnung von vorher, dem Lemma 1 und Theorem 5.7 in [1]. □

Korollar 2 *Netze in Netzen sind streng ausdrucksschwächer als Turingmaschinen.*

Der Beweis ergibt sich aus dem Fakt, dass das Halteproblem für Turingmaschinen nicht entscheidbar ist. □

Theorem 2. *Die Instandhaltbarkeit und Unvermeidbarkeit (w.r.t. $\leq$) sind entscheidbar für Netze in Netzen.*

Der Beweis folgt aus der Annahme zur Quasi-Ordnung von vorher, dem Lemma 1 und Theorem 5.9 in [1]. □

6 Abschließende Anmerkungen

Netze in Netzen sind eine Erweiterung der Petri-Netz Formalismen, die dem System eine visuelle, klar dynamische Hierarchie und modulare Struktur geben. Die Synchronisation von hierarchischen Komponenten ist vorgesehen und mächtig.

Die Struktur von Netzen in Netzen gibt eine gute Vorstellung von ihrem verteilten Verhalten. Obwohl wir hier nur eine verschränkte Semantik definiert haben, kann sie per Design auf simultane oder unabhängige Ausführungen verallgemeinert werden. Mit den beiden Arten der Synchronisation, horizontale Synchronisation für eine Zusammenarbeit von Elementen des Netzes und vertikale Synchronisation für die Koordination von Systemen und seinen Elementen, können Nested Petri Nets Formalismen als eine Art von Verallgemeinerung von modularen Petri Netzen und hierarchischen Petri Netzen angesehen werden.

Wie sich gezeigt hat, sind Netze in Netzen visuelle und ausdrucksstarke Werkzeuge für die Modellierung von Mehrbenutzer (Multi-Agent) verteilten Systemen. Gleichzeitig ist die Entscheidbarkeit für Eigenschaften, wie das Halteproblem eine wichtige Grundlage für diverse Verfikationsprobleme. Weniger ausdrucksstark als Turing Maschinen, behalten Netze in Netzen die Vorzüge von Petri Netzen.

7 Einsatzkontext Medizin - Erweiterte Workflow Netze

Die Motivation beschreibt bereits einen möglichen Einsatz im medizinischen Bereich. Kees van Hee et. al. beschreiben in der Veröffentlichung von 2006 den Einsatz von Nested Petri Nets beim modellieren von medizinischen Arbeitsprozessen. Dabei werden medizinische Vorgehensweisen und Protokolle bzw. Prozesse als Erweiterte Workflow-Netze modelliert und können dann genutzt werden, um den Behandlungsverlauf eines Patienten abzubilden.

7.1 Workflow-Netze

Traditionell werden zum Abbilden von Arbeitsprozessen Workflow-Netze genutzt. Um im späteren Verlauf des Berichts die Workflow-Netze zu erweitern, werden nun die Definitionen entsprechender Netze vorgestellt.

Definition 7.11 *Ein P/T-Netz $\mathcal{N}$ = (P,T,F,m_a) heißt Workflow-Netz (WF-Netz), falls gilt:*

- *es existieren zwei Plätze $\{a, e\} \in P$ mit ${}^\bullet a = \emptyset$ und $e^\bullet = \emptyset$*
- *alle Plätze und Transitionen auf Pfaden zwischen a und e liegen*
- *m_a durch [$m_a(p)$:= **if** $p = a$ **then** 1 **else** 0 **fi**] festgelegt ist*
- *m_e durch [$m_e(p)$:= **if** $p = e$ **then** 1 **else** 0 **fi**] festgelegt ist*

Die Bedingungen für die Korrektheit von Workflow-Netzen ist dabei besonders zu beachten, da diese aufgrund späterer Erweiterungen der Definition von Workflow-Netzen mit verändert wird.

Definition 7.12 *Ein WF-Netz $\mathcal{N}$ = (P,T,F,m_a) heißt korrekt, falls gilt:*

- $\forall m \in R(\mathcal{N}), \exists w \in T^* : m \xrightarrow{w} m_e$
- $\forall m \in R(\mathcal{N}) : m(e) \geq 1 \Rightarrow m = m_e$
- $\forall t \in T, \exists m \in R(\mathcal{N}) : m \xrightarrow{t}$

7.2 Medizinische Vorgehensweise - Prozessablauf

Die klassische Art medizinische Prozesse zu unterstützen ist das EPR Protokoll (Elecetronic Patient Record). Es handelt sich dabei um ein datenorientiertes System, indem für jeden Patienten eine Datenstruktur angelegt und in einer relationalen Datenbank gespeichert wird. Die Daten des Patienten werden dabei Schritt für Schritt im Verlauf des medizinischen Prozesses angepasst und erweitert. Aufgrund dieser Daten sind nun regelbasierte Systeme in der Lage die Entscheidungen des medizinischen Personals zu unterstützen. In den letzten Jahren ist die protkollbasierende Medizin zum Standardverfahren geworden. Dabei wurde eine große Mengen von Protokollen entwickelt, um Diagnosen zu Stellen und Behandlungen zu beschreiben. Bei diesen Protokollen handelt es sich um Prozessbeschreibungen, die der Mediziner Schritt für Schritt abarbeitet und Ergebnisse, sowie sein genaues Vorgehen dokumentiert. Dies ist nicht nur wichtig um die Qualität der Versorgung hoch zu halten sondern auch um Klarheit zu schaffen, falls etwas schief läuft.
Heutzutage haben die Protokolle die Form von Guidelines. Diese Guidlines sind nun nicht mehr nur auf Doktoren beschränkt und ziehen das gesamte Personal, wie Krankenschwestern und Sanitätern mit ein. Es gab verschiedenste Versuch dieses Guidelines zu formalisieren und beispielsweise als Flussdiagramme oder Entscheidungsdiagramme darzustellen, um diese in MDS-Systeme (Medical Decision Support Systeme) einzubinden.

7.3 Motivation: Erweiterung

Die nachfolgenden Erweiterungen der Workflow-Netze zum modellieren der soeben beschriebenen Prozessabläufen und Guidelines bringt besondere Vorteile mit sich. Im medizinischen Alltag findet sich eine Vielzahl von einzelnen Prozessen, die jeweils separat voneinander beschrieben werden können. Es kann also eine grundlegende Menge von Objektnetzen modelliert werden, welche dann in Systemnetzen zum Einsatz kommen. Auch diese Systemnetze können bei entsprechender Abstraktion wieder als Objektnetze ihrer dann zu modellierenden Systemnetze aufgefasst werden.Diese Vorgehensweise ist also ideal geeignet um aus einfachen Bausteinen wesentlich komplexere Protokolle zu schaffen.

Es werden in der folgenden Beschreibung vor allem die Ziele gesetzt, nicht nur die Markierungen eines Objektnetzes zu verändern, sondern auch die Objektnetze selber. Dies resultiert in folgenden Vorteilen:

- die Möglichkeit die Menge der Objektnetze zur Laufzeit zu ändern
- die Möglichkeit den Prozess zur Laufzeit zu ändern
- die Möglichkeit Entscheidungen von verschiedenen Parteien zu modellieren

7.4 Erweiterte Workflow-Netze

Die grundlegende Neuerung, die in den Erweiterten Workflow-Netzen eingeführt werden sind die sogenannten Ausnahmetransitionen. Diese Menge von Transitionen symbolisieren einen Ausnahmefall, welcher eintreten kann. Wenn erweiterte Workflow-Netze später als Objektnetze in Nested Petri-Nets genutzt werden lassen sich diese Ausnahmetransitionen nutzen um ein synchronisiertes Schalten mit ihren Systemnetzen zu realisieren.

Bevor wir die Erweiterten Workflow-Netze jedoch einführen können, müssen wir uns zunächst noch einmal mit der Labelfunktion für Transitionen in gefärbten Netzen beschäftigen. Um sicherzustellen, dass Ausnahmetransitionen auch als solche erkannt werden können und nicht mit normalen Transitionen verwechselt werden können, müssen wir die Menge der Transitionslabel Σ in zwei Teilmengen aufteilen. Dabei gibt Σ^e die Menge der Label für die Ausnahmetransitionen, Σ^n die Label für normale Transitionen an. Dabei gilt: $\Sigma = \Sigma^e \cup \Sigma^n$ und $\emptyset = \Sigma^e \cap \Sigma^n$. Nun lassen sich Erweiterte Workflow-Netze wie folgt definieren:

Definition 7.41 *Ein gefärbtes Petrinetz $N = (P, T, F, \upsilon, \rho, l)$ über dem Universum $\mathcal{U}$ ist ein Erweitertes Workflow Netz (EWF Netz) mit dem initialen Platz $a \in P$ und dem finalen Platz $e \in P$ und einer Menge von Ausnahmetransitionen $T' \subseteq T$ falls:*

- $\{t \mid (t,a) \in F\} = \{t \mid (e,t) \in F\} = \emptyset$;
- $\upsilon(a) = \upsilon(e) = \{black\}$;
- $\forall t \in T$ *and* $(\gamma, \delta) \in \rho(t)$, $t \in T'$ *gdw.* $l(t,\gamma,\delta) \in \Sigma^e$ *gdw.* $\{p \mid (t,p) \in F\} = \emptyset$;

– $\forall n,\ n \in (P \cup T)$ *gibt es einen Pfad von a nach n;*
– $\forall n,\ n \in (P \cup T)$ *gibt es einen Pfad von n zu m, mit* $m \in T' \cup \{e\}$.

Bei normalen Workflow-Netzen handelt es sich demnach um Erweiterte Workflow-Netze, in der die Menge von Ausnahmetransitionen leer ist.

7.5 Korrektheit von EWF-Netzen

Das Problem welches sich nun ergibt, ist dass die Definition der Korrektheit von Workflow-Netzen bei den Erweiterten Workflow-Netzen nicht mehr stimmt. Dies lässt sich leicht nachvollziehen, da für eine Markierung m $\in \mathcal{R}$(N), für die gilt: m $\xrightarrow{t}$ mit t $\in$ T* die Ausnahmetransition t aktiviert wäre. Da jedoch kein p existiert, für das gilt (t,p) $\in$ F kann dieser Pfad über t nicht zur Makierung m_e führen. Wir müssen die Korrektheit von EWF-Netzen also definieren als:

Definition 7.51 *Ein EWF-Netz N = (P, T, F, υ, ρ, l) mit einem initialen Platz a und einem finalen Platz e über dem Universum* $\mathcal{U}$ *wird korrekt genannt gdw.* $\forall (N, M) \in \mathcal{R}(N, [a])$

1. *entweder (N,M)* $\xrightarrow{*}$ *(N, [e])*
   oder $\exists (N, M') \in \mathcal{R}(N, M)$*, sodass* $(N, M') \xrightarrow{\sigma}$ *für ein* $\sigma \in \Sigma^e$
2. *(N,M)* $\xrightarrow{*} (N, m + [e]) \Rightarrow m = \emptyset$ *für ein* $m \in \mu(N)$.

Dabei ist besonders der zweite Abschnitt interessant. In Abbildung 6 ist das linke EWF-Netz nicht korrekt, da beim schalten der Transition t eine Marke in r und in q abgelegt wird. Wenn nun die Ausnahmetransition v schaltet, existiert im Netz auf q noch eine Marke, was den zweiten Punkt der Korrektheit der Definition verletzt. Das andere Netz erfüllt die Definition, da bei einer Marke in r$'$ nur entweder die Ausnahmetransition v$'$ oder die Transition u$'$ schalten kann.

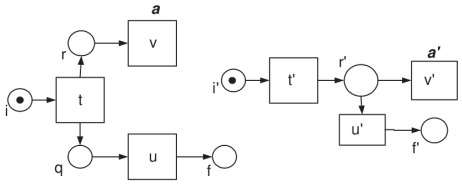

Abbildung 6. Links: Nicht Korrektes EWF-Netz, Rechts: Korrektes EWF-Netz

7.6 Eigenschaftenvon EWF-Netzen

Für die zukünftige Betrachtung von EWF-Netzen sind noch einige Eigenschaften zu nennen. Dabei wollen wir die Eigenschaft initial, final und Initialisierung

betrachten.

Definition 7.61 *Gegeben sein ein EWF-Netz N mit dem initialen Platz a und dem finalen Platz e und sei $m \in \mathcal{R}(N)$, dann nennt man das Netz N mit der Markierung m initial, bzw. final gdw. $m = m_a$ bzw. $m = m_e$ gilt.*
Die Initialisierung eines Netzes N wird mit Init(N) notiert und erzeugt ein Netz N mit der Anfangsmakierung m_a.

7.7 Verknüpfungen auf EWF-Netzen

Auf EWF-Netze können verschiedene Operationen angewandt werden. Diese ermöglichen beispielsweise die Hintereinanderausführung oder die parallele Ausführung von EWF-Netzen.

Sequenzen Um eine Sequenz von zwei EWF-Netzen zu bilden, werden diese mit dem Operator $\cdot$ verknüpft. Da jedes EWF-Netz eine initialen und eine finalen Platz besitzt, können zwei EWF-Netzen N_1 und N_2 mit den initialen Plätzen i_1, i_2 und den finalen Plätzen f_1, f_2 zusammen geschlossen werden, indem f_1 und i_2 gleich gesetzt werden. Somit ist die sequenzielle Komposition $N_1 \cdot N_2$ geschaffen worden. Ein Beispiel dafür findet sich in Abbildung 7.7.
Ein besonderer Fall, wie in den Beispielen zum Einsatz im medizinischen Sektor zu finden war, ist die Iteration, also die n-fache Hintereinanderausführung eines Netzes N. Dies wird als N^n notiert.

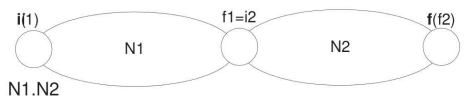

Abbildung 7. Hintereinanderausführung von EWF-Netzen

Parallelität Um eine Parallelität von zwei EWF-Netzen zu Bilden werden diese mit dem Operator $\parallel$ verknüpft. Zwei EWF-Netze N_1 und N_2 mit den Markierungen M_1 und M_2 werden dabei zu $(N_1 \parallel N_2, M_1 + M_2)$. In Abbildung 8 wird ein Beispiel für die Parallelausführung angegeben.

7.8 Einfache Protokolle

Im Beispiel wird die Behandlung von SCLC (Small-Cell-Lung-Cancer) als Motivation genutzt. Das Beispiel wird dabei von dem Guidlines inspiriert, die vom National Comprehensive Cancer Network (SUA) unter Modifizierung der Universität Texas M.D. Anderson Cancer Center erstellt wurden. Diese Guidlines

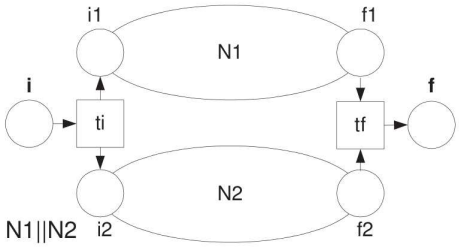

Abbildung 8. Paralleles Ausführen von EWF-Netzen

basieren auf einer Sammlung von Standartprotokollen. Ein Protokoll kann dabei den Prozess zum begründen einer Diagnose, ein Behandlungsschema oder ein Test während der Diagnosefindung bzw. nach einer Behandlung sein.
Jedes solche Protokoll hat einen Anfangspunkt, einen Endpunkt und es behandelt Ausnahmefälle durch das beenden des Prozesses. Als Beispiel lässt sich hier das obligatorische Anfangstestprotokoll in Abbildung 9 betrachten. Es besitzt einen Anfangspunkt p_b und einen Endpunkt p_f, sowie zwei Ausnahmetransitionen (*positive*). Eine Ausnahme wird ausgelöst, wenn mindestens einer der Tests als positiv bewertet wird. Notation der Protokolle: Ein Protokoll wird zukünftig

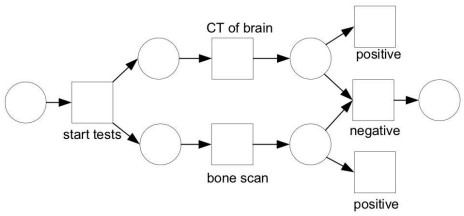

Abbildung 9. Mandatory Test Protocol (MandT<pos>

mit der Notation $Prot\langle ex_1, \cdots, ex_n \rangle$ aufgeführt, wobei Prot den Protokollnamen und $ex_1, \cdots, ex_n$ die Liste der Ausnahmelabels beschreibt. Falls ein Protokoll keine Ausnahmelabel besitzt, wird es lediglich durch seinen Namen definiert. Protkolle können andere Protokolle benutzten, neue Protokolle aus existierenden erzeugen oder verändern.

7.9 Zusammenführen von Protokollen

Wie im speziellen ein solches Netz mit mehreren Ebenen funktionieren kann und welche Logik dahinter steht zeigt sich in Abbildung 10. Dort werden folgende Dinge abgebildet:

Initialisierung Beim Erzeugen von *Surveillance* wird das Protokoll in Abbildung 10 erzeugt und es schaltet die Transistion *start surveillance*, dessen Ausgangskanten Nestet Nets erzeugen. Auf dem Platz p_1 wird ein Netz *Radiotherapy* · *STest* erzeugt. Auf dem Platz p_2 wird das Netz RadCon$\langle$*radiationscarring*$\rangle$ erzeugt.

Fall 2a: Das Protokoll RadCon überprüft im Verlauf der Radiotherapy, ob ein *Radiation Scarring* auftritt. Im Falle dessen wird eine Ausnahmetransition im RadCon Protokoll geschaltet. Da diese mit dem Systemnetz - in diesem Falle Surveillance - synchronisiert ist, schaltet auch die Transition im Systemnetz. An die Variable st wird das Objektnetz vom Platz p1 gebunden und die Ausgangskante der Transition RadiationScarring im Systemnetz legt dieses Objektnetz auch erneut auf den Platz p1 zurück, jedoch wird dazu Parallel das Protokoll ProphT (Prophylactic Treatment) erzeugt. Die Ausgangskante r müsste, da das RadCon Protokoll durch die Ausnahmetransition terminiert ist, erneut ein RadCon Protokoll erzeugen. Dazu wird im Beispiel jedoch keine nähere Erläuterung gegeben.

Fall 2b: Wir nehmen zunächst an, dass kein RadiationScarring auftritt. Das Objektnetz auf dem Paltz p_1 wird also nach und nach ausgeführt und erreicht zu einem bestimmten Punkt den STest der am Ende die drei Transitionen *relapse*, *partial response* und *good response* aufweist. Diese sind erneut mit dem Systemnetz vertikal synchronisiert.
Im Falle der Transition *relapse* schaltet das Systemnetz also und belegt die Variable r mit dem RadCon Objektnetz.
Im Falle der Transition *good response* wird über die Ausgangskante im Systemnetz erneut das Protokoll *Radiotherapy* · *STest* erzeugt und das entsprechende Objektnetz auf dem Platz p_1 gelegt. Im Falle der Transition *partial response* wird eine andere Kombination von Protokollen gewählt. Es wird parallel zur Radiotherapy eine Chemotherapy eingesetzt, die danach mit einem STest abgeschlossen wird (init(((Cisplatin · Etoposide) ‖ Radiotherapy) · STest)).

Termiantion

Sobald die Transition *relapse* geschaltet wurde, ist das Surveillance Protokoll final. Es befindet sich in der finalen Markierung und das Protokoll ist abgeschlossen.

7.10 Komplettes SCLC-Protokoll

Das komplette Protokoll zur Vorgehensweise bei SCLC findet sich in Abbildung 11 und gibt einen Überblick über die jeweiligen Schritte. Bei der Transition ganz

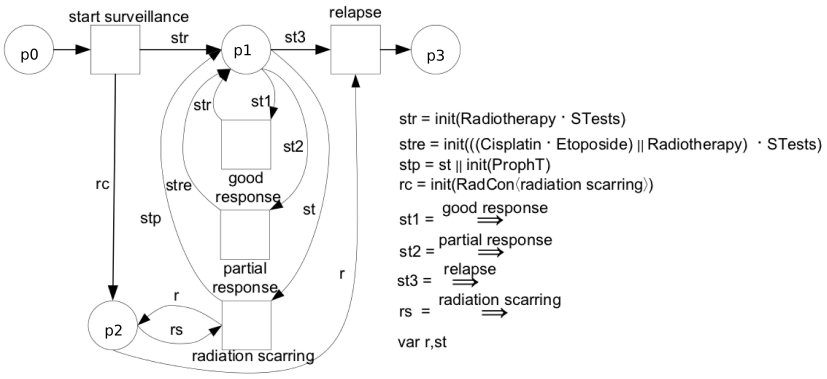

Abbildung 10. Surveillance Protocol (Surveillance)

unten, links lässt sich die Kantenbeschriftung *sv* erkennen, die das Initialisieren des Surveillance Protokolls beschreibt, was soeben vorgestellt wurde. Auch lässt sich in diesem Protokoll der MandatoryTest$\langle positive \rangle$ wiederfinden. Ein besonderer Punkt ist noch einmal beim Protokoll FTest$\langle operable, non-operable \rangle$ zu beobachten. Dieses Netz besitzt lediglich Ausnahmetransitionen um den Übergang zum nächsten Zustand zu realisieren. Auch dies ist mit EWF-Netzen also durchaus möglich.

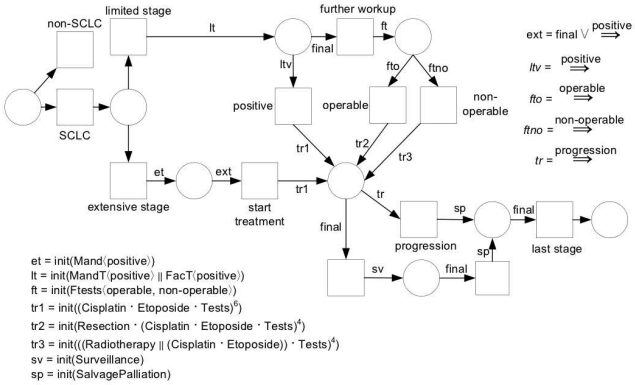

Abbildung 11. SCLC Protocol

7.11 Bewertung

Allgemein lassen sich die EWF-Netze als durchaus sinnvoll beim Einsatz im medizinischen Sektor bewerten. Da gerade das Arbeitsfeld in der Medizin aus vielen kleinen Teilschritten besteht, die sich sehr gut modellieren und besonders durch Petri-Netze sehr gut visualisieren lassen, liefert dieser Ansatz eine viel versprechende Lösung. Besonders interessant ist dabei natürlich die gleichzeitige Visualisierung, Formalisierung und Verifizierbarkeit. Es lassen sich Arbeitsprozess, die gerade im medizinischen Bereich dramatische Auswirkugnen haben können auf Fehler und Schwachstellen überprüfen.

Literaturverzeichnis

1. Irina A. Lomazova and Ph. Schnoebelen. Some decidability results for nested petri nets. In *Proceedings of the Third International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, PSI '99, pages 208–220, London, UK, 2000. Springer-Verlag.
2. Kees M. van Hee, Irina A. Lomazova, Olivia Oanea, Alexander Serebrenik, Natalia Sidorova, and Marc Voorhoeve. Nested nets for adaptive systems. In *ICATPN*, pages 241–260, 2006.