6.4 Fairness: 5-Philosophen

Fairness ist eine wichtige Erscheinungen der Verifikation von Systemen. Fairness ist verwandt mit Lebendigkeit und ist im Zusammenhang mit temporallogischen Spezifikationen wichtig.

Zum Begriff der Fairness betrachten wir das Problem der fünf Philosophen [Dij75], mit dem ein Betriebsmittelzuteilungsproblem besonderer Art beschrieben wird.

Fünf Philosophen $Ph_1, \ldots, Ph_5$ sitzen an einem runden Tisch, in dessen Mitte eine Schüssel mit Spaghetti steht (Abb. 6.28). Jeder Philosoph Ph_i befindet sich entweder im Zustand des „Denkens“ (d_i) oder „Essens“ (e_i). Zum Essen stehen insgesamt nur 5 Gabeln $g_1, \ldots, g_5$ (die Betriebsmittel) zur Verfügung, jeweils eine zwischen zwei benachbarten Philosophen. Geht ein Philosoph vom Denken zum Essen über, nimmt er erst die rechte und dann die linke Gabel auf.

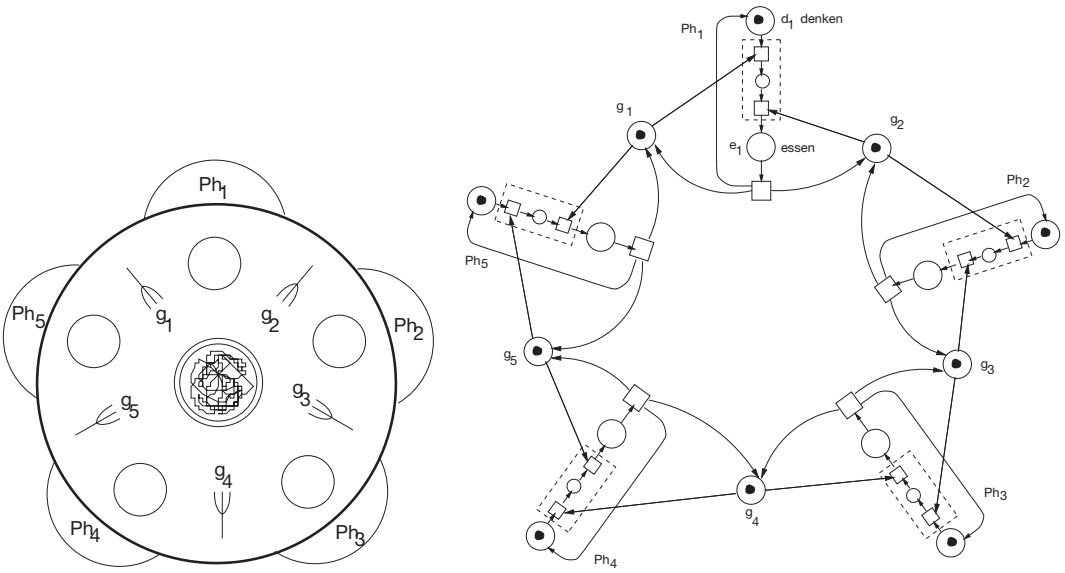


Abbildung 6.28: Die fünf Philosophen am Tisch (nach Dijkstra) und als P/T-Netz

Abbildung 6.28 zeigt eine P/T-Netz-Darstellung des Problems. Das Netz ist natürlich nicht lebendig: Wenn alle fünf Philosophen ihre rechte Gabel nehmen, entsteht eine Verklemmung. Um dies zu verhindern, kann man die beiden Transitionen, die das Aufnehmen der rechten und linken Gabel darstellen, unteilbar machen, also zu der in Abb. 6.28 dargestellten Vergröberung übergehen. Nun ist das Netz zwar lebendig, aber es besteht immer noch die Möglichkeit, dass zwei Philosophen, etwa Ph_1 und Ph_3 so die Gabel benutzen, dass Ph_2 nie die Chance hat, seine Gabeln aufzunehmen. Man sagt, für den Philosophen Ph_2 besteht die Gefahr des Verhungerns (*starvation*), oder die Philosophen Ph_1 und Ph_3 verhalten sich unfair gegenüber Ph_2.

Definition 6.34 *Ein P/T-Netz* $\mathcal{N} = (S, T, F, W, \mathbf{m}_0)$ *hat ein* faires Verhalten*, oder verhält sich* fair *(behaves fairly), wenn in jeder unendlichen Schaltfolge* $w \in F_\omega(\mathcal{N})$ *jede Transition* $t \in T$ *unendlich oft vorkommt.*
Dabei sei $F_\omega(\mathcal{N}) := \{w = a_1 a_2 a_3 \cdots \in T^\omega \mid \forall i \geq 1 : a_1 a_2 \cdots a_i \in FS(\mathcal{N})\}$.

Zur Definition von $FS(\mathcal{N})$ siehe Definition 6.14 auf Seite 100.
Wir vergleichen die wichtigen Begriffe der Lebendigkeit und Fairness:

a) Lebendigkeit bedeutet Freiheit von *unvermeidbaren* partiellen Verklemmungen.
b) Fairness bedeutet Freiheit von *faktischen* partiellen Verklemmungen.

Der Unterschied zwischen lebendigem und fairem Verhalten ist also gekennzeichnet durch den Existenzquantor in a) (alle Transitionen *können* immer wieder schalten) und dem Allquantor in b) (alle Transitionen *müssen* immer wieder schalten).
Außer in einfachen Fällen hat ein System oder Netz kein faires Verhalten. In dem Netz von Abb. 6.29 kann natürlich die faire Folge schalten:

$$acbd \quad acbd \quad \ldots,$$

aber auch die unfaire:

$$ac \quad ac \quad ac \quad ac \quad \ldots.$$

Dieses Netz entspricht in gewisser Weise dem folgenden Programm:

do a $\rightarrow$ c
$\Box$ b $\rightarrow$ d **od**

Hierbei sind c und d Anweisungen, die a und b nicht verändern und letztere mit **true** initialisiert sind. Die **do**...**od** - Klammer bezeichnet eine Schleifenanweisung mit nichtdeterministischer Ausführung des Schleifenkörpers und geschützten Anweisungen (*guarded commands*).

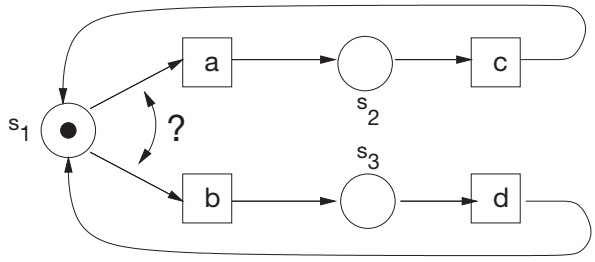


Abbildung 6.29: Netz mit unfairem Verhalten

Sowohl für die Netze als auch für Programme hat man daher Formen des fairen Schaltens bzw. Programmablauf vorgeschlagen. Das Problem wird bei der Definition von Programmiersprachen heute jedoch meist noch dem Implementator zugeschrieben.

Definition 6.35 *Es sei $\mathcal{N} = (P, T, F, W, \mathbf{m}_0)$ ein P/T-Netz und $t \in T$.*

a) t schaltet produktiv *oder* verschleppungsfrei *oder* nach der verschleppungsfreien Schaltregel *(finite delay property), wenn*

$$\forall \mathbf{m} \in \mathbf{R}(\mathcal{N})\ \neg\exists w \in T^\omega : \mathbf{m} \xrightarrow{w} \wedge t \text{ ist bei } w \text{ permanent aktiviert } \wedge |w|_t = 0$$

b) t schaltet fair*, oder* nach der fairen Schaltregel *(fair firing rule) , wenn*

$$\forall \mathbf{m} \in \mathbf{R}(\mathcal{N})\ \neg\exists w \in T^\omega : \mathbf{m} \xrightarrow{w} \wedge\ t \text{ ist bei } w \text{ unendlich oft aktiviert } \wedge |w|_t = 0$$

c) Das Netz $\mathcal{N}$ schaltet nach der produktiven bzw. fairen Schaltregel, wenn dies alle seine Transitionen tun.

Ein Netz schaltet also nicht verschleppungsfrei (bzw. fair), wenn von einer erreichten Markierung $\mathbf{m}$ ab eine unendliche Schaltfolge w schaltet, bei der eine Transition zwar permanent, d.h. in allen durchlaufenen Markierungen (bzw. unendlich oft) aktiviert ist, aber nie schaltet.

Zu implementieren wäre diese Eigenschaft etwa durch Zähler, die über die Aktiviertheit von Transitionen Buch führen. Ab einer festgelegten Größe des Zählers würde dieser Transition dann Priorität eingeräumt.

Die verschleppungsfreie Schaltregel ist die elementarere. Sie sorgt z.B. dafür, dass unabhängige nebenläufige Anweisungen nach endlicher Zeit ausgeführt werden. Die faire Schaltregel wird oft in der Semantik nichtdeterministischer oder nebenläufiger Programme aufgenommen. Das folgende Programm terminiert z.B. zwingend nur unter der fairen Schaltregel:

```
b := true;
 do   b  →  x := 1
   □  b  →  b := false   od
```

Anhand der vorstehenden Beispiele kann man die Beziehung zwischen verschleppungsfreiem bzw. fairem Schalten und lebendigem bzw. fairem Verhalten studieren.

Beispiel 6.36

a) Das 5-Philosophen-Netz von Abb. 6.28 ist zwar lebendig, hat aber auch unter der verschleppungsfreien Schaltregel kein faires Verhalten ($ac\ ac\ \ldots$ ist immer noch möglich). Das Netz verhält sich jedoch fair bei der fairen Schaltregel.

b) Das 5-Philosophen-Netz von Abb. 6.28 mit der vergröberten, und daher unteilbaren Transition ist lebendig. Es hat aber weder unter der verschleppungsfreien noch unter der fairen Schaltregel ein faires Verhalten.

c) Das 5-Philosophen-Netz von Abb.6.28 ohne Vergröberung ist nicht lebendig. Unter der fairen Schaltregel hat es jedoch ein faires Verhalten.

d) Verhindert man die Verklemmung durch andere Maßnahmen, z.B. indem man höchstens vier Philosophen in den Essraum lässt (Abb. 6.30), dann ist das Netz lebendig und fair bei der fairen Schaltregel.

   In der dargestellten Anfangsmarkierung befinden sich alle Philosophen im Nebenraum. Die Stelle s_i bewirkt, dass höchstens eine Marke nach d_i und die Stelle h, dass höchstens vier Marken nach d_1 bis d_5 gelangen.

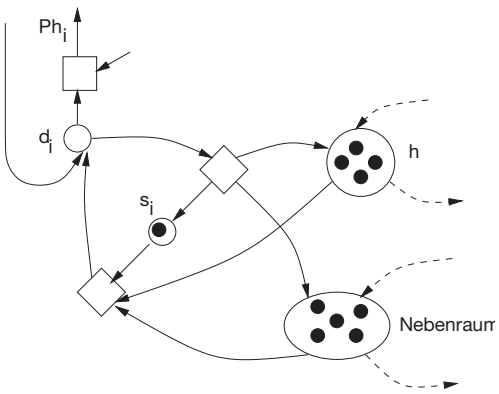


Abbildung 6.30: Philosophenproblem mit Nebenraum

6.5 Fortschritt und Fairness

Das Netz in Abbildung 6.31 a) erlaubt die Wiederholung der Aktionen a und b, d.h. *abababab* ... ist eine aktivierte Schaltfolge. Für jeden Zustand, der während dieser Schaltfolge erreicht wird, ist die Transition c aktiviert, ohne je zu schalten. Um zu erzwingen, dass c schaltet, müssen wir (in der Schaltfolgen-Semantik) die produktive Schaltregel (Definition 6.35) anwenden. In der Prozess-Semantik stellt sich dies in viel natürlicherer Weise als Fortschrittseigenschaft dar.

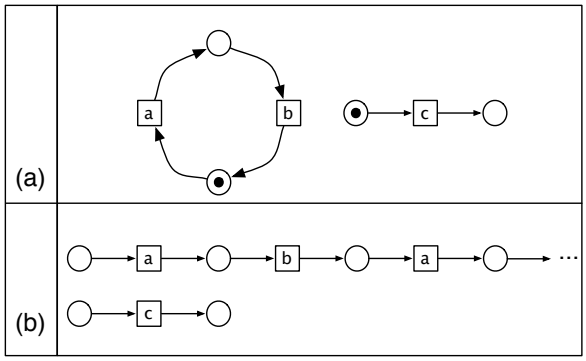


Abbildung 6.31: Zur Fortschrittseigenschaft

Definition 6.37 *Sei ein P/T-Netz N gegeben. Ein Prozess (R, ϕ) mit (B, E, F_K) respektiert die* Fortschrittseigenschaft *(engl. progress) einer Teilmenge der Transition $T^p \subseteq T$, wenn der maximale Schnitt R° nur Transitionen aus $T \setminus T^p$ aktiviert:*

$$\forall e \in E \quad \forall p \in P : W(p, \phi_T(e)) = |\phi_P^{-1}(p) \cap R^\circ| \Rightarrow t \notin T^p$$

Eine Transition $t \in T^p$ heißt *Fortschrittstransition*. Positiv formuliert bedeutet Fortschritt, dass eine dauerhaft aktivierte Fortschrittstransition $t \in T^p$ auch schalten muss. Abbildung 6.31 b) zeigt einen Prozess (hier der einzige maximale Prozess) des Netzes aus Abbildung 6.31 a). Falls c eine Fortschrittstransition ist, d.h. $c \in T^p$, muss sie in jedem Prozess vorkommen.
Die produktive Schaltregel (Definition 6.35) in der Schaltfolgen-Semantik entspricht also dem Fall $T = T^p$ als *Fortschrittseigenschaft*.

Eigenschaft 6.38 *Wenn N ein lebendiges Netz ist und für alle Transitionen die Fortschrittseigenschaft gilt, dann sind alle Prozesse, welche die Fortschrittseigenschaft respektieren, unendlich.*

7 Systemverifikation durch Petrinetze

Es existieren verschiedene Verifikationsmethoden für Petrinetze:

a) enumerative Methoden: Der Erreichbarkeitsgraph wird analysiert (normaler Weise nur bei beschränkten Netzen möglich).
b) Transformationen: Das Netz wird – unter Beibehaltung der untersuchten Eigenschaft – vereinfacht (Reduktion).
c) strukturelle Analyse: Hierbei wird die Netzstruktur herangezogen, um Eigenschaften – unabhängig von der Initialmarkierung – nachzuweisen. Dies geschieht u.a. durch Methoden der linearen Algebra (P-Invarianten, T-Invarianten) oder durch graphenbasierte Methoden.

7.1 Verifikation beschränkter Netze

Ist das Netzsystem $(\mathcal{N}, \mathbf{m_0})$ beschränkt, dann ist $\mathbf{R}(\mathcal{N}, \mathbf{m_0})$ endlich und kann konstruiert werden. In diesem Fall sind die in diesem Skript behandelten Methoden des Model-Checking anwendbar, falls die zu prüfende Eigenschaft in temporaler Logik spezifiziert ist. Wir wählen hier einen elementareren Ansatz, indem wir die Eigenschaften direkt im Zustandsgraphen (das ist der Erreichbarkeitsgraph) formulieren.

7.1.1 Grundlagen

Der Algorithmus 7.1 liefert eine Berechnung dieses Graphen. Die Kennzeichnung der Knoten (durch Färben) in Schritt 2.1 bewirkt, dass jede Markierung nur einmal bearbeitet wird. In Schritt 2.2.3 werden nur solche Markierungen zu V hinzugefügt, die noch nicht vorhanden sind.
Die Konstruktion des Erreichbarkeitsgraphen ist sehr aufwendig, da er im Verhältnis zur Größe des Netzes exponentiell viele Knoten enthalten kann (siehe [Val92]). Darauf werden wir im Abschnitt 7.3 zurückkommen.
Der Erreichbarkeitsgraph $\mathrm{RG}((\mathcal{N}, \mathbf{m_0}))$ eines unbeschränkten Netzsystems $(\mathcal{N}, \mathbf{m_0})$ ist nicht endlich. Karp and Miller [KM69] haben bewiesen, wie man dies durch die Bedingung feststellen kann, die in Schritt 2.2.2 von Algorithmus 7.1 als Abbruchkriterium dient. Dies wird in folgendem Satz 7.1 präzisiert.

Algorithmus 7.1 (Berechnung des Erreichbarkeitsgraphen)

Input - Das Netzsystem $(\mathcal{N}, \mathbf{m_0})$
Output - Der gerichtete Graph $\mathrm{RG}(\mathcal{N}, \mathbf{m_0}) = (V, E)$, falls das Netzsystem beschränkt ist.

1. Initialisiere $\mathrm{RG}(\mathcal{N}, \mathbf{m_0}) = (\{\mathbf{m_0}\}, \emptyset)$; $\mathbf{m_0}$ sei ungefärbt;
2. **while** Es gibt ungefärbte Knoten in V. **do**
   - 2.1 Wähle einen ungefärbte Knoten $\mathbf{m} \in V$ und färbe ihn.
   - 2.2 **for** Für jede in $\mathbf{m}$ aktivierte Transition t **do**
     - 2.2.1 Berechne $\mathbf{m}'$ mit $\mathbf{m} \xrightarrow{t} \mathbf{m}'$;
     - 2.2.2 **if** Es gibt einen Knoten $\mathbf{m}''' \in V$ derart, dass $\mathbf{m}''' \xrightarrow{\sigma} \mathbf{m}'$ und $\mathbf{m}''' < \mathbf{m}'$.
       then Der Algorithmus terminiert ohne Ergebnis.;
       (Das Netzsystem ist unbeschränkt.)
     - 2.2.3 **if** Es gibt keinen Knoten $\mathbf{m}''' \in V$ derart, dass $\mathbf{m}''' = \mathbf{m}'$
       then $V := V \cup \{\mathbf{m}'\}$, wobei $\mathbf{m}'$ ein ungefärbter Knoten sei.
     - 2.2.4 $E := E \cup \{(\mathbf{m}, t, \mathbf{m}')\}$
3. Der Algorithmus terminiert mit Ergebnis. ($\mathrm{RG}(\mathcal{N}, \mathbf{m_0})$ ist der Erreichbarkeitsgraph.)

Satz 7.1 *Ein Netzsystem $\mathcal{N}$ ist genau dann unbeschränkt, wenn es zwei erreichbare Markierungen $\mathbf{m}_1, \mathbf{m}_2 \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0)$ gibt, die folgende Bedingungen erfüllen:*

a) $\exists \sigma \in T^* : \mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}_2$

b) $\mathbf{m}_1 \lneq \mathbf{m}_2$

Beweis: Gilt die Bedingung, dann kann σ auch in $\mathbf{m}_2$ schalten, da ja mindestens so viele Marken wie in $\mathbf{m}_1$ vorhanden sind: $\mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}_2 \xrightarrow{\sigma} \mathbf{m}_3$. Außerdem gilt $\mathbf{m}_2 - \mathbf{m}_1 = \mathbf{m}_3 - \mathbf{m}_2$ und damit $\mathbf{m}_2 \lneq \mathbf{m}_3$. Also gilt die Bedingung auch für das Paar $\mathbf{m}_2, \mathbf{m}_3$ und man erhält per Induktion $\mathbf{m}_1 \xrightarrow{\sigma} \mathbf{m}_2 \xrightarrow{\sigma} \mathbf{m}_3 \xrightarrow{\sigma} \mathbf{m}_4 \xrightarrow{\sigma} \mathbf{m}_5 \xrightarrow{\sigma} .....$
Wegen $\mathbf{m}_1 \lneq \mathbf{m}_2 \lneq \mathbf{m}_3 \lneq \mathbf{m}_4 \lneq \mathbf{m}_5 \lneq ....$ ist das System unbeschränkt.
Ist umgekehrt der Erreichbarkeitsgraph unendlich, dann gibt es nach dem Lemma von König eine unendliche Folge $\mathbf{m}_1 \xrightarrow{*} \mathbf{m}_2 \xrightarrow{*} \mathbf{m}_3 \xrightarrow{*} \mathbf{m}_4 \xrightarrow{*} \mathbf{m}_5 \xrightarrow{*} ....$ mit $\mathbf{m}_1 \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0)$ und $\mathbf{m}_1 \lneq \mathbf{m}_2 \lneq \mathbf{m}_3 \lneq \mathbf{m}_4 \lneq \mathbf{m}_5 \lneq .....$ Also ist die Bedingung erfüllt. □

Theorem 7.2 *Folgende Aussagen sind für gegebenes $(N, \mathbf{m}_0)$ äquivalent:*

1. *Die Menge der erreichbaren Markierungen $R(N, \mathbf{m}_0)$ ist endlich.*
2. *Der Erreichbarkeitsgraph $RG(N, \mathbf{m}_0)$ ist endlich.*
3. *$\mathcal{N}$ ist in der Markierung $\mathbf{m}_0$ beschränkt.*
4. *Es gibt keine Markierungen $\mathbf{m}_1, \mathbf{m}_2$ mit $\mathbf{m}_2 \gneq \mathbf{m}_1$ und $\mathbf{m}_0 \xrightarrow{*} \mathbf{m}_1 \xrightarrow{*} \mathbf{m}_2$.*

Beweis: Als Übung.

□

Auch vergleichsweise kleine Komponenten eines System müssen rechnergestützt verifiziert werden. Daher hat das Forschungsgebiet *rechnergestützte Verifikation* (computer-aided verification) große praktische Bedeutung. Es folgt als Beispiel ein kleines Netzsystem, das nur schwer ohne Rechnerunterstützung zu analysieren ist.

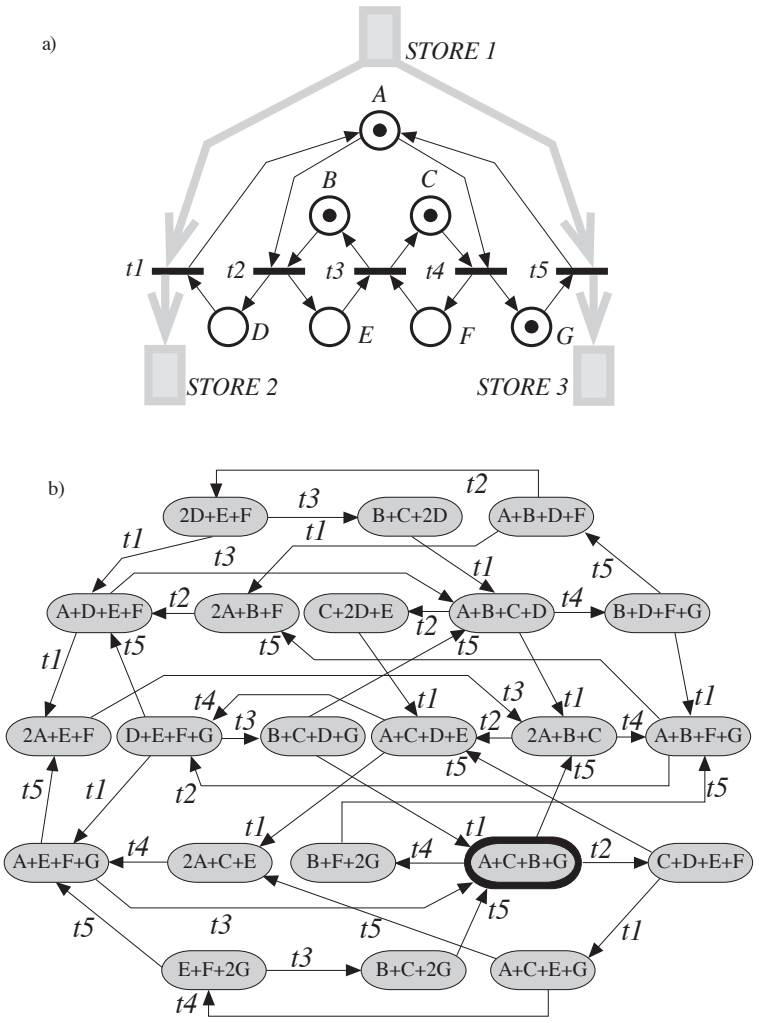


Abbildung 7.1: Ein kleines Netzsystem (a) mit komplexem Erreichbarkeitsgraph in (b)

Beispiel 7.3 Abb. 7.1 zeigt ein einfaches Netzsystem: Teile einer Produktionsanlage sollen von *STORE 1* zu *STORE 2* oder *STORE 3* transportiert werden. Das durch die Plätze $\{B, C, E, F\}$ erzeugte Teilnetz spezifiziert eine Regel, nach der die Teile auf die Lager zu verteilen sind. Der angegebene Erreichbarkeitsgraph ist schwer zu analysieren, obwohl er „strukturiert“ dargestellt ist. Man prüfe mit ihm, dass folgende Regel implementiert wurde: die Teile werden gleichmäßig auf beide Lager verteilt, es darf aber bis zu 4 konsekutive Zuteilungen zu einem Speicher geben, was langfristig wieder auszugleichen ist.

7.1.2 Markierungs-Invarianzeigenschaft

Es wird nun ein Verfahren beschrieben, das Systemeigenschaften aus zwei Klassen in polynomieller Zeit (im Verhältnis zur Größe des Erreichbarkeitsgraphen) entscheidet. Die Systemeigenschaften werden durch *Markierungsprädikate* Π spezifiziert.
Markierungsprädikate sind aussagenlogische Formeln, deren Atome Ungleichungen der Form:

$$\Pi(\mathbf{m}) = \left[\sum_{p \in P} k_p \mathbf{m}(p) \leq k\right] \qquad k_p, k \in \mathbb{Q}$$

Beispiel: $\Pi(\mathbf{m}) = (2 \cdot \mathbf{m}(p_1) + \mathbf{m}(p_2) \ \leq \ 3 \vee \mathbf{m}(p_3) \leq 1)$
Beispiel: $\Pi(\mathbf{m}) = \mathbf{m}(p_1) = 1$ (Gleichheit = denke man sich hier mittels $\leq$ ausgedrückt.)
Die erste Klasse von Spezifikationen heißt *Markierungs-Invarianzeigenschaften* (engl. marking invariance property).

Definition 7.4 *Sei* Π *ein Markierungsprädikat.*
Eine Markierungs-Invarianzeigenschaft *eines Netzsystems* $(\mathcal{N}, \mathbf{m_0})$ *ist ein Prädikat der Form:*

- $\forall\, \mathbf{m} \in \mathbf{R}(\mathbf{m_0}) : \Pi(\mathbf{m})$ *oder*
- $\forall\, \mathbf{m} \in \mathbf{R}(\mathbf{m_0})\ \exists\, t \in T : \Pi(\mathbf{m})$

Beispiele dazu sind:

1) *k-Beschränktheit* (k-boundedness) eines Platzes p:

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m}_0)\,.\,\mathbf{m}(p) \leq k$$

2) *Markierungs-Ausschluss* (marking mutual exclusion) zwischen p und p':

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m}_0)\,.\,((\mathbf{m}(p) = 0) \vee (\mathbf{m}(p') = 0))$$

3) *Verklemmungsfreiheit* (deadlock-freeness):

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m}_0)\,.\,\exists t \in T\,.\,W(\bullet, t) \leq \mathbf{m}$$

Algorithmus 7.2 (Entscheiden einer Markierungs-Invarianzeigenschaft)

Input - Der Erreichbarkeitsgraph $\mathrm{RG}(\mathcal{N}, \mathbf{m_0})$; die Markierungs-Invarianzeigenschaft Π.
Output - TRUE falls die Eigenschaft Π erfüllt ist; FALSE falls die Eigenschaft Π nicht erfüllt ist.

1. Initialisiere alle Elemente von $\mathbf{R}(\mathcal{N}, \mathbf{m_0})$ als ungefärbt.
2. **while** Es gibt einen ungefärbten Knoten $\mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m_0})$ **do**
   2.1 Wähle einen ungefärbten Knoten $\mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m_0})$ und färbe ihn.
   2.2 **if** $\mathbf{m}$ erfüllt nicht Π.
   then return FALSE (Die Eigenschaft Π ist nicht erfüllt.)
3. Return TRUE

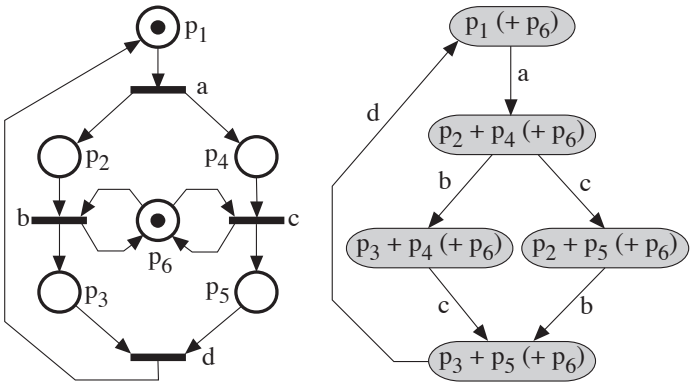


Abbildung 7.2: Beschränktes, lebendiges und reversibles Netz mit Erreichbarkeitsgraph

Markierungs-Invarianzeigenschaften können mit Algorithmus 7.2 entschieden werden, dessen Zeitkomplexität linear in Bezug auf die Größe von $\mathbf{R}(\mathcal{N}, \mathbf{m}_0)$ ist, denn jeder Knoten wird höchstens einmal aufgesucht. Falls der Algorithmus erfolgreich terminiert, dann erfüllen alle von $\mathbf{m_0}$ aus erreichbaren Markierungen das Prädikat Π. Falls der Algorithmus in Schritt 2.2 terminiert, dann gibt es einen Pfad in $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0)$, der von $\mathbf{m_0}$ ausgeht und in einer Markierung endet, die Π nicht erfüllt.

Beispiel 7.5 Wir wollen für das Netz in Abb. 7.2 testen, ob Markierungsausschluss der Plätze p_5 und p_6 gilt:

$$\Pi(\mathbf{m}) = (\mathbf{m}(p_5) = 0) \vee (\mathbf{m}(p_6) = 0)$$

Die Anwendung des Algorithmus 7.2 zur Überprüfung von $\Pi(\mathbf{m})$ beginnt damit, dass alle Elemente von $\mathbf{R}(\mathbf{m_0})$ ungefärbt sind (Schritt 1).
Dann werden die Markierungen nacheinander geprüft bis $p_2 + p_5 + p_6$ erreicht wird. Hier wird das Prädikat Π als ungültig festgestellt und der Algorithmus terminiert mit FALSE.

7.1.3 Lebendigkeits-Invarianzeigenschaften

Die zweite Klasse von Spezifikationen heißt *Lebendigkeits-Invarianzeigenschaften* (engl. liveness invariance property).

Definition 7.6 *Sei* Π *ein Markierungsprädikat.*
Eine Lebendigkeits-Invarianzeigenschaft *eines Netzsystems* $(\mathcal{N}, \mathbf{m_0})$ *ist ein Prädikat der Form*

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m_0}) \,.\, \exists \mathbf{m}' \in \mathbf{R}(\mathbf{m}) \,.\, \Pi(\mathbf{m}')$$

Beispiele dazu sind:

1) *Lebendigkeit von* t (liveness of t):

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m}_0) \,.\, \exists \mathbf{m}' \in \mathbf{R}(\mathbf{m}) \,.\, W(\bullet, t) \leq \mathbf{m}'$$

2) $\mathbf{m}_H$ ist *Rücksetzzustand* (home state):

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m_0}) \,.\, \exists \mathbf{m}' \in \mathbf{R}(\mathbf{m}) \,.\, \mathbf{m}' = \mathbf{m}_H$$

3) *Reversibilität* (reversibility):

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m_0}) \,.\, \exists \mathbf{m}' \in \mathbf{R}(\mathbf{m}) \,.\, \mathbf{m}' = \mathbf{m_0}$$

Diese Eigenschaften lassen sich nicht wie in Algorithmus 7.2 durch lineares Durchsuchen der Erreichbarkeitsmenge überprüfen. Vielmehr muss von jeder erreichbaren Markierung eine von dieser erreichbare Markierung gefunden werden, die Π erfüllt.
Der im folgende entwickelte Algorithmus 7.3, der Lebendigkeits-Invarianzeigenschaften prüft, greift auch das Konzept der *strengen Zusammenhangskomponente* (strongly connected component) eines Graphen zurück.

Definition 7.7 *Ein* Pfad *von* $\mathbf{m}_1$ *nach* $\mathbf{m}_k$ *in einem Erreichbarkeitsgraphen* $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0) = (V, E)$ *ist eine Folge* $\mathbf{m}_1 \ldots \mathbf{m}_i \mathbf{m}_{i+1} \ldots \mathbf{m}_k$ $(k \geq 2)$ *seiner Knoten mit* $(\mathbf{m}_i, t_i, \mathbf{m}_{i+1}) \in E$ *für alle* $i \in \{1, \ldots, k-1\}$ *und jeweils ein* $t_i \in T$.
Ein Teilgraph von $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0)$ *heißt* streng zusammenhängend*, falls er entweder nur aus einem Knoten besteht oder für je zwei verschiedene seiner Knoten* $\mathbf{m}_i$ *und* $\mathbf{m}_j$ *ein Pfad von* $\mathbf{m}_i$ *nach* $\mathbf{m}_j$ und *von* $\mathbf{m}_j$ *nach* $\mathbf{m}_i$ *existiert.*
Seine Knotenmenge heißt Zusammenhangskomponente (ZK).
Eine maximale ZK heißt strenge Zusammenhangskomponente (SZK) *von* $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0)$.
Sie heißt triviale Zusammenhangskomponente*, falls sie nur aus einem Knoten ohne Schleife besteht.*
Sie heißt terminale strenge Zusammenhangskomponente*, falls von keinem ihrer Knoten eine Kante zu einem Knoten* $\mathbf{m}$ *ausgeht, der nicht in ihr liegt.*

Die strengen Zusammenhangskomponenten eines gerichteten Graphen (V, E) können in der Zeitkomplexität $O(|V| + |E|)$ berechnet werden (siehe [Meh84], Algorithmus von Tarjan).

Lemma 7.8 *Die Knotenmengen verschiedener SZKs sind disjunkt.*

Beweis: Als Übung. □

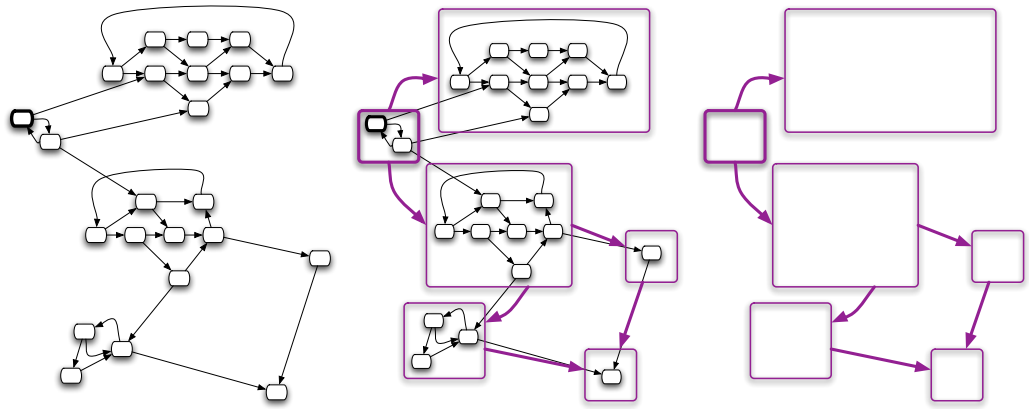
Da die Knotenmengen verschiedener SZKs disjunkt sind, kann man sie in eindeutiger Weise zu einem Knoten zusammenziehen. Wenn man die verbleibenden Kanten sinngemäß überträgt, erhält man seinen *reduzierten Graphen.*
Seine Kantenmenge E_c ist dann die Menge aller (C_i, t, C_j), so dass $C_i \neq C_j$ gilt (keine Schleifen) und es eine Kante $(\mathbf{m}_i, t, \mathbf{m}'_j) \in E$ gibt, wobei $\mathbf{m}_i$ in der SZK C_i und $\mathbf{m}'_j$ in der SZK C_j liegt.

Definition 7.9 *Der* reduzierte Graph $\mathrm{RG}^c(\mathcal{N}, \mathbf{m}_0) = (V_c, E_c)$ *eines Erreichbarkeitsgraphen* $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0) = (V, E)$ *besteht aus:*

- *Die SZKs von* $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0)$ *sind die Knotenmenge, d.h.* $V_c := \{C_1, \ldots, C_r\}$.
- *Die Kantenmenge ist* $E_c := \{(C_i, t, C_j) \mid \exists(\mathbf{m}_i, t, \mathbf{m}'_j) \in E : \mathbf{m}_i \in C_i \wedge \mathbf{m}'_j \in C_j \wedge C_i \neq C_j\}$.

Als Anfangsknoten wird der Knoten definiert, der aus derjenigen SZK entstanden ist, die den Anfangsknoten enthält (vorausgesetzt, der Ausgangsgraph hatte einen solchen).

Beispiel 7.10 Abb. 7.3 b) zeigt die strengen Zusammenhangskomponenten des Graphen von Abb. 7.3 a), von denen zwei terminal sind. Zwei seiner SZKs sind trivial.



a) b) c)

Abbildung 7.3: a) Ein Graph, b) seine SZKs und c) der reduzierte Graph

In Abb. 7.3 c) ist der reduzierte Graph von Abb. 7.3 a) dargestellt. Den beiden terminalen SZK entsprechend, hat der azyklische Graph zwei Knoten, von denen keine Kanten ausgehen.

Lemma 7.11 *Der reduzierte Graph* $\mathrm{RG}^c(\mathcal{N}, \mathbf{m}_0)$ *ist azyklisch.*

Beweis: Als Übung. □

Da der reduzierte Graph $\mathrm{RG}^c(\mathcal{N}, \mathbf{m}_0)$ immer azyklisch ist, können seine terminalen SZKs in linearer Zeitkomplexität berechnet werden. Dies wird im Algorithmus 7.3 zur Prüfung von Lebendigkeits-Invarianzeigenschaften ausgenutzt. Dieser hat daher auch eine lineare Zeitkomplexität (in Bezug auf die Größe von $\mathrm{RG}(\mathcal{N}, \mathbf{m}_0)$). Wenn der Algorithmus erfolgreich terminiert, enthalten alle terminalen SZKs eine Markierung, die die Eigenschaft Π erfüllt. Daher existiert zu jeder erreichbaren Markierung eine Nachfolgemarkierung, die Π erfüllt. Falls der Algorithmus nicht erfolgreich terminiert, enthält mindestens eine terminale SZK eine Markierung, die die Eigenschaft Π nicht erfüllt.
Als wichtigen Spezialfall erhalten wir:

Algorithmus 7.3 (Entscheiden einer Lebendigkeits-Invarianzeigenschaft)

Input - Der Erreichbarkeitsgraph $\mathrm{RG}(\mathcal{N}, \mathbf{m_0})$. Die Lebendigkeits-Invarianzeigenschaft Π.
Output - TRUE falls die Eigenschaft Π erfüllt ist; FALSE falls die Eigenschaft Π nicht erfüllt ist.

1. Berechne die strengen Zusammenhangskomponenten (SZKs) $C_1, \ldots, C_r$ von $\mathrm{RG}(\mathcal{N}, \mathbf{m_0})$.
2. Berechne den reduzierten Graphen $\mathrm{RG}^c(\mathcal{N}, \mathbf{m}_0) = (V_c, E_c)$ von $\mathrm{RG}(\mathcal{N}, \mathbf{m_0})$
3. Berechne die Menge F der terminalen SZKs von $\mathrm{RG}^c(\mathcal{N}, \mathbf{m}_0)$.
4. **while** es gibt $C_i \in F$ **do**
   4.1 **if** C_i enthält keine Markierung $\mathbf{m}'$, die Π erfüllt.
   then return FALSE
   4.2 Entferne C_i aus F
5. Return TRUE

Lemma 7.12 *Eine Transition $t \in T$ des Netzsystems $(\mathcal{N}, \mathbf{m}_0)$ ist genau dann lebendig, wenn t als Anschrift in jeder terminalen SZK des reduzierte Graphen $\mathrm{RG}^c(\mathcal{N}, \mathbf{m}_0)$ vorkommt.*

Beispiel 7.13 Analyse einer Lebendigkeits-Invarianzeigenschaft
Wir betrachten das Netzsystem von Abb.6.18 b) mit dem Erreichbarkeitsgraphen von Abb. 7.4. Um den Algorithmus 7.3 auszuführen, müssen zunächst die SZKs berechnet werden. Diese sind schon in Abb. 7.4 durch die Bereiche C_1, C_2 und C_3 eingezeichnet, wobei C_2 und C_3 terminal sind.
Wir prüfen die Lebendigkeitseigenschaft:

$$\forall \mathbf{m} \in \mathbf{R}(\mathbf{m_0}) \,.\, \forall t \in T \,.\, (\exists \mathbf{m}^t \in \mathbf{R}(\mathbf{m}) \,.\, \mathbf{m}^t \geq W(\bullet, t))$$

Dazu sei t eine feste Transition und $\Pi(\mathbf{m}) = (\mathbf{m} \geq W(\bullet, t))$.
Dann wird in Schritt 4 des Algorithmus festgestellt, dass jede der beiden terminalen SZK eine Markierung $\mathbf{m}$ enthält die Π erfüllt, d.h. dass $\Pi(\mathbf{m})$ gilt. Da dies für alle Transitionen erfolgreich durchgeführt werden kann, ist das Netzsystem lebendig.
Führt man Algorithmus 7.3 aus, um zu prüfen ob die Markierung $\mathbf{m}_H = p_2 + p_3 + p_6 + p_7 + p_9 + p_{14}$ ein Rücksetzzustand ist, erhält man das Ergebnis FALSE, da zwar die terminale SZK C_2 die Markierung $\mathbf{m}_H$ enthält, nicht jedoch C_3.

Anmerkung: Man kann effektivere Algorithmen entwickeln, wenn man besondere Charakteristiken der zu prüfenden Eigenschaft *oder* des zu analysierenden Systems kennt.
Als Beispiel für den ersteren Fall nehmen wir die Eigenschaft der Reversibilität. Dann müssen alle terminalen SZKs die Anfangsmarkierung enthalten, d.h. der Erreichbarkeitsgraph ist insgesamt streng zusammenhängend. Es genügt in diesem Fall also zu prüfen, dass es nur eine SZK gibt.
Zum zweiten Fall nehmen wir an, dass wir schon wissen, dass das System reversibel ist. Dann ist der Erreichbarkeitsgraph streng zusammenhängend. Um die Lebendigkeit einer Transition t zu analysieren, genügt es dann zu prüfen, ob t überhaupt an irgend einer Kante des Erreichbarkeitsgraphen steht.

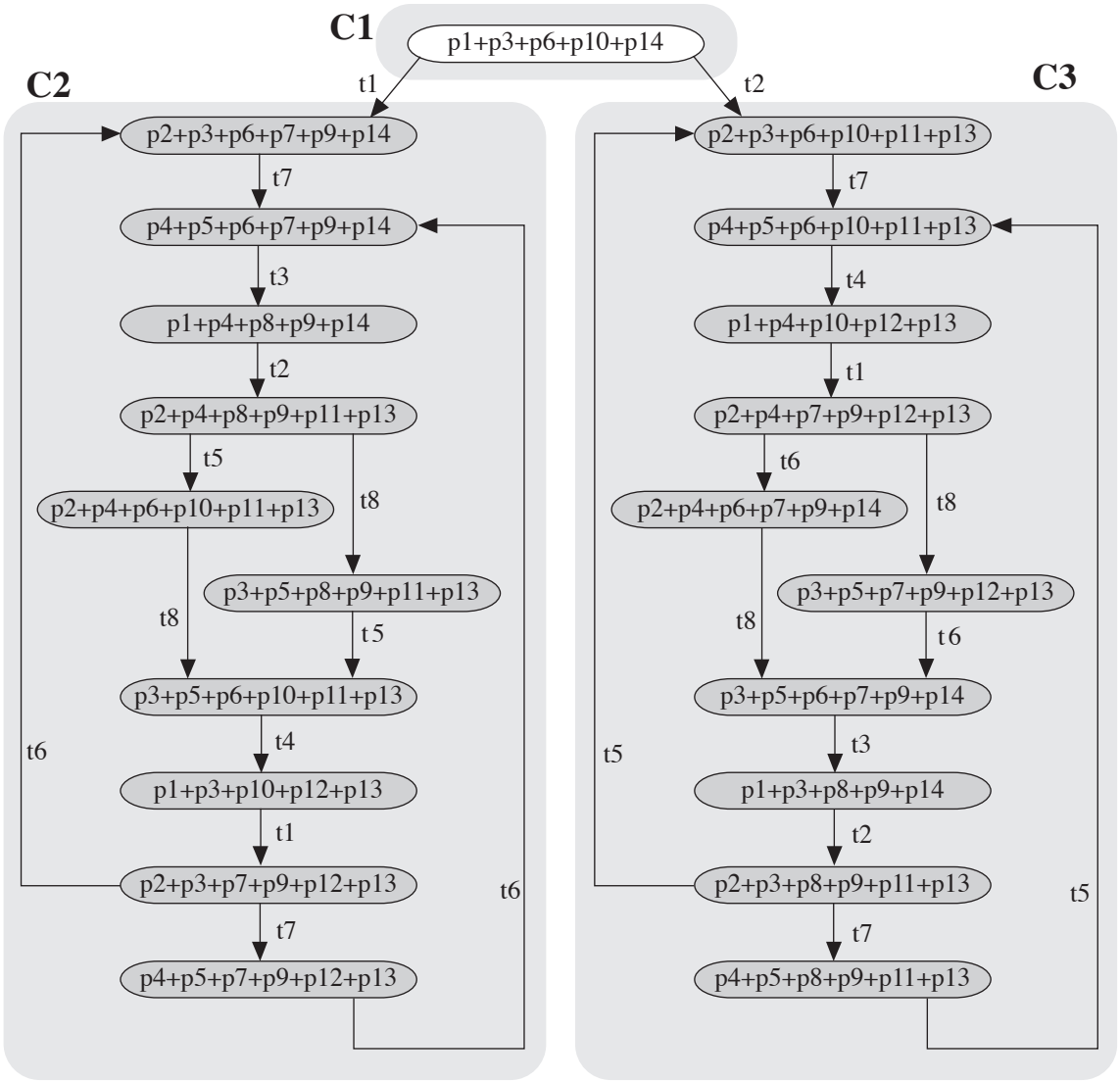


Abbildung 7.4: Erreichbarkeitsgraph des Netzsystems von Abb. 6.18.b

7.2 Der Überdeckungsgraph

Der Algorithmus 7.1 zur Berechnung des Erreichbarkeitsgraphen bricht ohne Ergebnis ab, wenn das Netz unbeschränkt ist. Das ist unbefriedigend, wenn man z.B. zur Fehleranalyse wissen muss, welche Plätze unbeschränkt sind oder wie der Erreichbarkeitsgraph in Teilen aussieht, wo sich die Unbeschränktheit nicht auswirkt.
Mit dem Symbol ω (Omega) wird im Folgenden die Möglichkeit beliebig hoher (aber natürlich endlicher) Markenanzahlen auf einem Platz bezeichnet. Dazu werden die natürlichen Zahlen mit diesem formalen Symbol erweitert, sowie *Pseudomarkierungen* eingeführt, deren Komponenten dieses Omega enthalten.

Definition 7.14 *Es sei* $\mathbb{N}_\omega := \mathbb{N} \cup \{\omega\}$ *zusammen mit folgenden Rechenregeln:*

$$\forall n \in \mathbb{N} : \omega > n;$$

$$\forall n \in \mathbb{N}_\omega : \omega + n = \omega - n := \omega;$$

$$\forall n \in \mathbb{N} \setminus \{0\} : n \cdot \omega = \omega \cdot n := \omega; 0 \cdot \omega = \omega \cdot 0 := 0$$

Ein Vektor $\mathbf{m} \in \mathbb{N}_\omega^P$ *wird* Pseudomarkierung *genannt, wenn in ihm das Symbol* ω *vorkommt, und für diese wird die Schaltregel formal übernommen. Eine Pseudomarkierung* $\mathbf{m}$ *entspricht einer gewöhnlichen (Teil-) Markierung auf den Plätzen* s *mit* $\mathbf{m}(s) \neq \omega$*, wobei die mit* ω *besetzten Komponenten beliebig sind und unberücksichtigt bleiben.*
Für Vektoren $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{Z}^r$ *seien die Operatoren* $+, -, =$ *jeweils komponentenweise erklärt.*
Die partielle Ordnung $\leq$ *ergibt sich komponentenweise:* $\mathbf{m}_1 \leq \mathbf{m}_2$*, falls* $\forall p \in P : \mathbf{m}_1(p) \leq \mathbf{m}_2(p)$*.*
Die strikte Ordnung $\mathbf{m}_1 < \mathbf{m}_2$ *(besser:* $\mathbf{m}_1 \lneq \mathbf{m}_2$*) steht für* ($\mathbf{m}_1 \leq \mathbf{m}_2$ *und* $\mathbf{m}_1 \neq \mathbf{m}_2$*).*

7.2.1 Konstruktion des Überdeckungsgraphen

Wir definieren im Folgenden den Überdeckungsgraphen. Ein Überdeckungsgraphen ist eine reduzierte Version des Erreichbarkeitsgraphen. Es wird sich zeigen, dass ein *Überdeckungsgraph* stets endlich ist, während dies bei Erreichbarkeitsgraphen bekanntermaßen nicht immer der Fall ist. Somit ist es unvermeidbar, dass ein Überdeckungsgraph weniger Information über die Zustände enthält als der Erreichbarkeitsgraph. Wie wir aber sehen werden, reicht diese eingeschränkte Version bereits aus, um einige (wenn auch nicht alle) Fragen zu Petrinetzen zu entscheiden – insbesondere die Fragestellung, ob ein Netz beschränkt ist.
Sei $\mathcal{N} = (P, T, F, W, \mathbf{m}_0)$ ein P/T-Netz. Wir konstruieren durch Algorithmus 7.4 einen gerichteten, kantenbeschrifteten Graphen $G(\mathcal{N}) := (V, E)$ mit $V \subseteq \mathbb{N}_\omega^P$ und Kanten $E \subseteq V \times T \times V$ und nennen ihn *Überdeckungsgraph*. Dabei schreiben wir $\mathbf{m}_1 \xrightarrow[w]{*} \mathbf{m}_2$, wenn es in $G(\mathcal{N})$ einen Pfad vom Knoten $\mathbf{m}_1$ zum Knoten $\mathbf{m}_2$ gibt, der die in der

Algorithmus 7.4 (Berechnung eines Überdeckungsgraphen)

Input - Das P/T-Netz $\mathcal{N} = (P, T, F, W, \mathbf{m}_0)$
Output - Der Überdeckungsgraph Graph $G(\mathcal{N}) = (V, E)$.

1. Initialisiere $G(\mathcal{N}) = (\{\mathbf{m_0}\}, \emptyset)$; $\mathbf{m_0}$ sei ungefärbt;
2. **while** Es gibt ungefärbte Knoten in V **do**
   2.1 Wähle einen ungefärbte Knoten $\mathbf{m} \in V$ und färbe ihn.
   2.2 **for** Für jede in $\mathbf{m}$ aktivierte Transition t **do**
   2.2.1 Berechne $\mathbf{m}'$ mit $\mathbf{m} \xrightarrow{t} \mathbf{m}'$ und $X(\mathbf{m}') := \{\mathbf{m}''' \in V | \mathbf{m}''' \leq \mathbf{m}' \text{ und } \mathbf{m}''' \xrightarrow{*} \mathbf{m}'\}$;
   2.2.2 **if** $X(\mathbf{m}') \neq \emptyset$ **then** $\mathbf{m}_1(p) := \begin{cases} \omega, & \exists \mathbf{m}''' \in X(\mathbf{m}') : \mathbf{m}'''(p) < \mathbf{m}'(p) \\ \mathbf{m}'(p), & \text{sonst.} \end{cases}$
   else $\mathbf{m}_1 := \mathbf{m}'$;
   2.2.3 **if** $\mathbf{m}_1 \notin V$
   then $V := V \cup \{\mathbf{m}_1\}$, wobei $\mathbf{m}_1$ ein ungefärbter Knoten sei. ;
   2.2.4 $E := E \cup \{(\mathbf{m}, t, \mathbf{m}_1)\}$;
3. Der Algorithmus terminiert mit Ergebnis. ($G(\mathcal{N})$ ist der Überdeckungsgraph.)

Kantenfolge zusammengefügte Beschriftung $w \in T^*$ hat. Wir schreiben $\mathbf{m}_1 \xrightarrow{*} \mathbf{m}_2$, wenn es irgendeinen Pfad gibt, dessen Beschriftung uns nicht wichtig ist.

Beispiel 7.15 Abb. 7.5 zeigt ein P/T-Netz $\mathcal{N}$ mit Überdeckungsgraph.

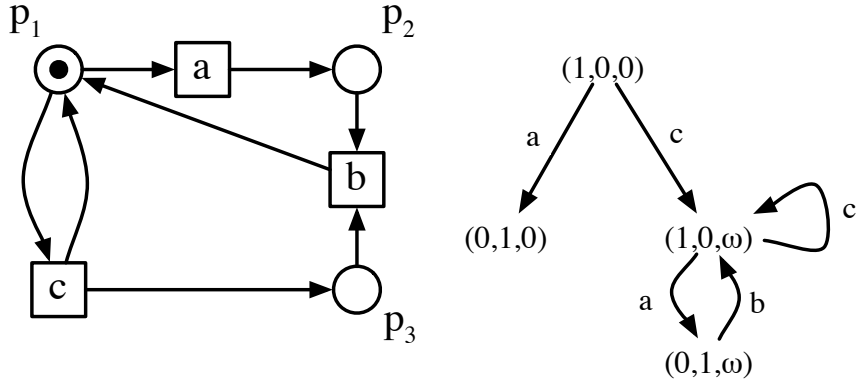


Abbildung 7.5: Ein Netz $\mathcal{N}$ mit Überdeckungsgraph

Die Bedeutung des Überdeckungsgraphen ergibt sich aus folgenden Eigenschaften:

Satz 7.16 *Sei $\mathcal{N} = (P, T, F, W, \mathbf{m}_0)$ ein P/T-Netz und $G(\mathcal{N}) = (V, E)$ ein Überdeckungsgraph zu $\mathcal{N}$, dann ist ein Platz $p \in P$ genau dann* beschränkt, *wenn es keinen Knoten $\mathbf{m} \in V$ mit $\mathbf{m}(p) = \omega$ gibt.*
Gilt $\mathbf{m}_0 \xrightarrow{w} \mathbf{m}$ im Netz $\mathcal{N}$ für ein Wort $w \in T^*$, so gibt es in $G(\mathcal{N})$ einen Knoten $\mathbf{m}_1$ mit $\mathbf{m}_1 \geq \mathbf{m}$ und $\mathbf{m}_0 \xrightarrow[w]{*} \mathbf{m}_1$.

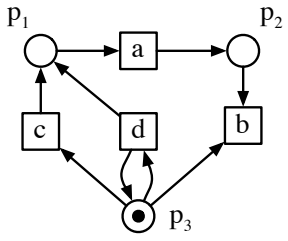
Aus dieser letzten Eigenschaft leitet sich der Name „Überdeckungsgraph“ für $G(\mathcal{N})$ ab, denn zu jedem in $\mathcal{N}$ erreichbaren Knoten $\mathbf{m} \in \mathrm{RG}(\mathcal{N})$ gibt es einen, i.A. anderen, in

$G(\mathcal{N})$, der diesen „überdeckt“. Um diesen Sachverhalt zu beweisen, zeigen wir zunächst die Termination dieses Algorithmus.

Satz 7.17 *Der Algorithmus 7.4 zur Konstruktion von $G(\mathcal{N})$ terminiert.*

Beweis: Angenommen, der Algorithmus terminiere *nicht*, dann ist $|V|$ unendlich.
Begründung: Wäre $|V|$ endlich, so auch die Menge $V \times T$ und alle Paare $(m,t) \in (V \times T)$ wären einmal nach Eintritt in die **while**–Schleife ausgewählt worden und die Termination wird erreicht.
Nun sind nach Konstruktion alle Knoten $\mathbf{m} \in V$ von $\mathbf{m}_0$ aus erreichbar. Betrachten wir den spannenden Baum $B(\mathcal{N})$ von $G(\mathcal{N})$, der dadurch entsteht, dass diejenigen Kanten weggelassen werden, die im Schritt 2.2.4 zu einem schon existierenden Knoten gezeichnet werden. $B(\mathcal{N})$ entsteht also aus $G(\mathcal{N})$ durch Streichen gewisser Kanten.
Da $B(\mathcal{N})$ verzweigungsendlich ist (jeden Knoten verlassen maximal $|T|$ viele Kanten) aber selbst unendlich viele Knoten besitzt, gibt es in $B(\mathcal{N})$ nach dem Satz von König (1936) einen unendlichen Pfad $\mathbf{m}_0 \to \mathbf{m}_1 \to \mathbf{m}_2 \to \ldots \quad \ldots \to \mathbf{m}_i \to \mathbf{m}_{i+1} \to \ldots$ in dem kein Knoten $\mathbf{m}_i$ zweimal vor kommt.
In jeder unendlichen Folge von paarweise verschiedenen Vektoren $\mathbf{m}_i \in \mathbb{N}^P$ gibt es, nach einem Satz von Dickson (1926), eine unendliche Teilfolge $\mathbf{m}_{i_1} \overset{+}{\to} \mathbf{m}_{i_2} \overset{+}{\to} \mathbf{m}_{i_3} \overset{+}{\to} \ldots \overset{+}{\to} \mathbf{m}_{i_j} \overset{+}{\to} \mathbf{m}_{i_{j+1}} \overset{+}{\to} \ldots$ mit der Eigenschaft $\mathbf{m}_{i_j} < \mathbf{m}_{i_{j+1}}$.
Nach Konstruktion muss $\mathbf{m}_{i_{j+1}}$ eine ω-Komponente mehr als $\mathbf{m}_{i_j}$ haben, denn $\mathbf{m}_{i_j} \overset{*}{\underset{B(\mathcal{N})}{\to}} \mathbf{m}_{i_{j+1}}$ impliziert $\mathbf{m}_{i_j} \overset{*}{\underset{G(\mathcal{N})}{\to}} \mathbf{m}_{i_{j+1}}$ und wenn $\mathbf{m}_{i_j} < \mathbf{m}_{i_{j+1}}$ ist, wird $\mathbf{m}_{i_{j+1}}$ mindestens eine ω-Komponente mehr enthalten als $\mathbf{m}_{i_j}$. Dies führt zu dem gewünschten Widerspruch, denn nur endlich viele ω-Komponenten sind überhaupt möglich. Also terminiert die Konstruktion von $G(\mathcal{N})$ nach dem Algorithmus 7.4. □

Aufgabe 7.18 Konstruieren Sie einen Überdeckungsgraphen $G(\mathcal{N})$ für das folgende P/T-Netz $\mathcal{N}$.



Satz 7.19 *Für ein P/T-Netz $\mathcal{N}$ ist* $\mathrm{RG}(\mathcal{N})$ *genau dann endlich, wenn kein Knoten von $G(\mathcal{N})$ eine ω-Komponente besitzt.*

Beweis: (kein $\omega \Rightarrow \mathrm{RG}(\mathcal{N})$ ist endlich)

Kommt in keiner Komponente von Knoten aus V in $G(\mathcal{N}) = (V, E)$ die Bezeichnung ω vor, so ist $G(\mathcal{N})$ identisch mit dem Erreichbarkeitsgraph von $\mathcal{N}$, und, wegen der Termination des Verfahrens, ist auch die Menge $V = \mathbf{R}(\mathcal{N})$ endlich.
($\omega \Rightarrow \mathrm{RG}(\mathcal{N})$ ist unendlich)
Sei $\mathbf{m}_2 \in V$ mit $\mathbf{m}_2(s) = \omega$ für ein $s \in P$, so gibt es einen Pfad in $G(N)$

$$\mathbf{m}_0 \xrightarrow[\alpha]{*} \mathbf{m}_{3,s} \xrightarrow[\beta]{*} \mathbf{m}_s \xrightarrow[t]{} \mathbf{m}_{1,s} \xrightarrow[\gamma]{*} \mathbf{m}_{2,s}$$

und $\mathbf{m}_{3,s} < \mathbf{m}_{1,s}$, genauer: $\mathbf{m}_{3,s}(s) < \mathbf{m}_{1,s}(s)$.
Da das Wort w auf dem Pfad von $\mathbf{m}_0$ über $\mathbf{m}_{3,s}$ nach $\mathbf{m}_s$ eine Schaltfolge im Netz $\mathcal{N}$ ist, gibt es also Wörter $\alpha, \beta, \gamma \in T^*$ und erreichbare Markierungen $\tilde{\mathbf{m}}_3, \tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_4, \ldots$ mit $\mathbf{m}_0 \xrightarrow{\alpha} \tilde{\mathbf{m}}_3 \xrightarrow{\beta t} \tilde{\mathbf{m}}_1 \xrightarrow{\gamma} \tilde{\mathbf{m}}_4 \ldots$.
Da $\tilde{\mathbf{m}}_3(s) < \tilde{\mathbf{m}}_1(s) < \tilde{\mathbf{m}}_4(s) < \ldots$ gilt, ist die Platz $s \in P$ also in $\mathcal{N}$ nicht beschränkt und $\mathbf{R}(\mathcal{N})$ ist keine endliche Menge. □

Wir erhalten aus den vorangegangenen Resultaten insbesondere die folgende Aussage.

Korollar 7.20 *Das* Beschränktheitsproblem *ist entscheidbar.*

7.2.2 Aussagekraft des Überdeckungsgraphen

Jede Schaltfolge w des Netzes besitzt einen gleichbeschriebenen Pfad w im Überdeckungsgraph. Die Umkehrung gilt im Allgemeinen jedoch nicht, wie das folgende Netz in Abbildung 7.6 zeigt. Das Netz hat den folgenden Erreichbarkeitsgraph:

$$\binom{1}{0} \underset{b}{\overset{a}{\rightleftarrows}} \binom{1}{1} \underset{b}{\overset{a}{\rightleftarrows}} \binom{1}{2} \cdots \underset{b}{\overset{a}{\rightleftarrows}} \binom{1}{n} \underset{b}{\overset{a}{\rightleftarrows}} \cdots$$

und den folgenden Überdeckungsgraph:

$$\binom{1}{0} \xrightarrow{a} \binom{1}{\omega} \circlearrowright a, b$$

Man beachte, dass es keine mit b beschriftete Kante von $\binom{1}{\omega}$ nach $\binom{1}{0}$ gibt. Nun ist beispielsweise $w = abb$ ein Kantenzug im Überdeckungsgraph, dieser ist aber nicht in in $\mathbf{m}_0$ aktiviert.

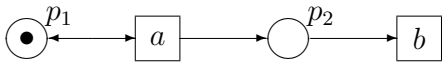


Abbildung 7.6: Beispielnetz

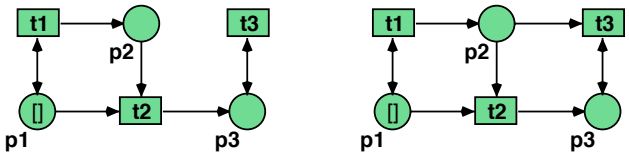
Der Überdeckungsgraph liefert somit die Möglichkeit, Nichterreichbarkeit einer Markierung nachzuweisen, denn Nicht-Überdeckbarkeit impliziert Nicht-Erreichbarkeit. Allerdings ist Erreichbarkeit nicht auf diese Art und Weise nachweisbar und auch nicht Lebendigkeit.

Theorem 7.21 *Es ist mit Hilfe des Überdeckungsgraphen nicht entscheidbar,*

1. *ob eine Transition lebendig ist,*
2. *ob ein Netz lebendig ist, oder*
3. *ob eine Markierung* $\mathbf{m}'$ *erreichbar ist.*

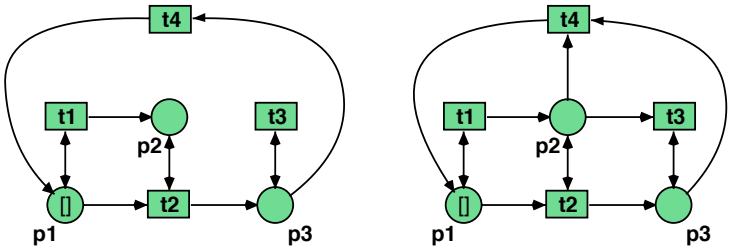
Beweis: Dies zeigen wir durch die Angabe zweier Netzpaare mit gleichem Überdeckungsgraph, bei dem ein Netz die Eigenschaft besitzt, das andere aber nicht.

1. Im linken Netz ist t_3 lebendig, im rechten aber nicht.



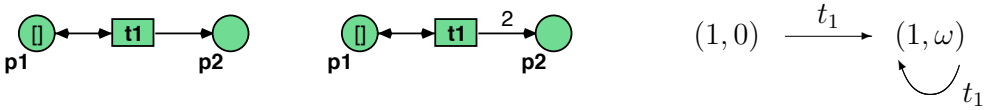
$$(1,0,0) \xrightarrow{t_1} (1,\omega,0) \xrightarrow{t_2} (0,\omega,1), \quad t_1 \text{ an } (1,\omega,0), \quad t_3 \text{ an } (0,\omega,1)$$

2. Das linke Netz ist lebendig, das rechte ist nach der Schaltfolge $t_1t_2t_3$ tot.



$$(1,0,0) \xrightarrow{t_1} (1,\omega,0) \underset{t_4}{\overset{t_2}{\rightleftarrows}} (0,\omega,1), \quad t_1 \text{ an } (1,\omega,0), \quad t_3 \text{ an } (0,\omega,1)$$

3. Im linken Netz ist die Markierung $(1,1)^{tr}$ erreichbar, im rechten dagegen nicht.



Also sind die Eigenschaften nicht mittels Überdeckungsgraphen entscheidbar. □