

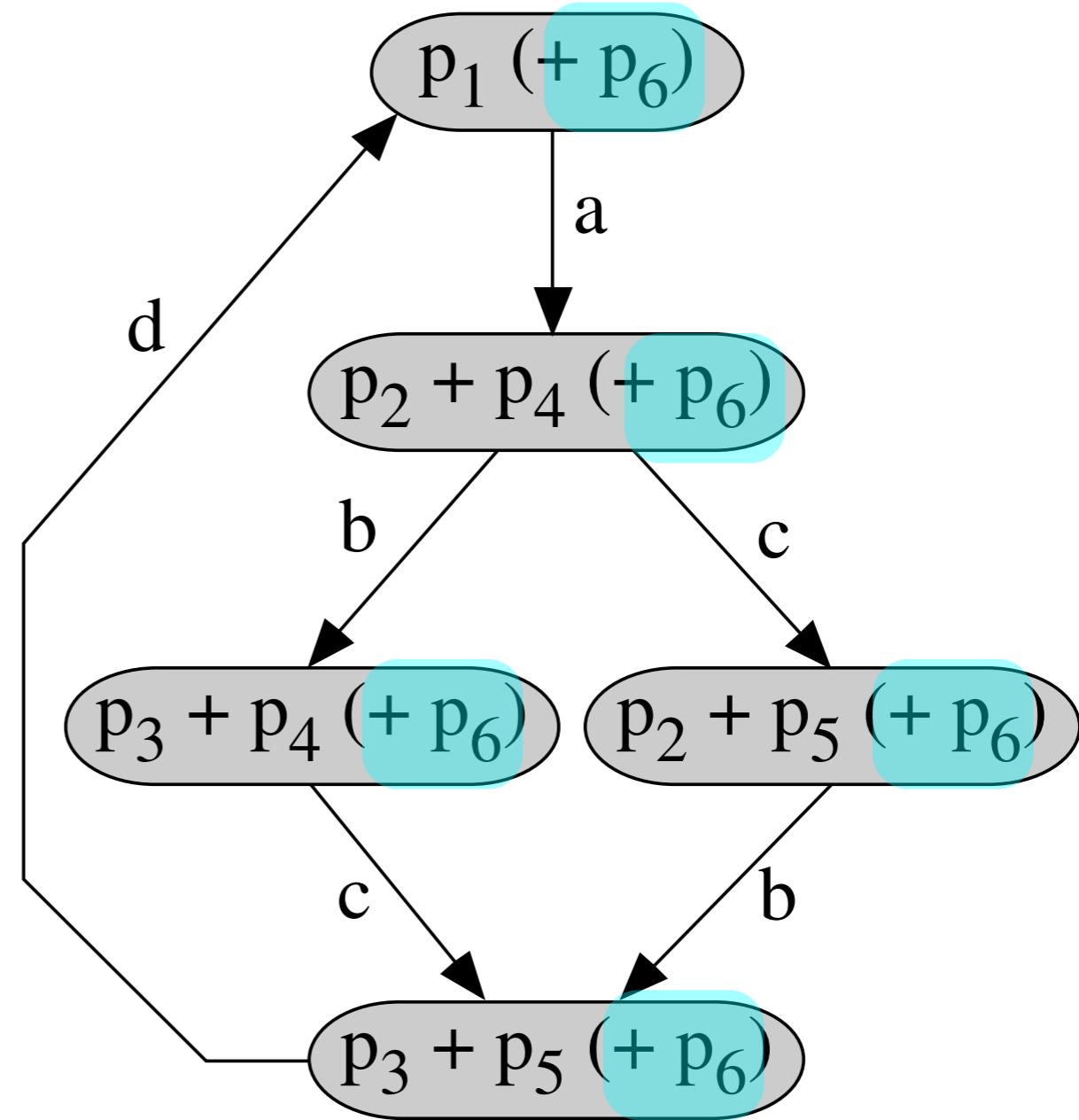
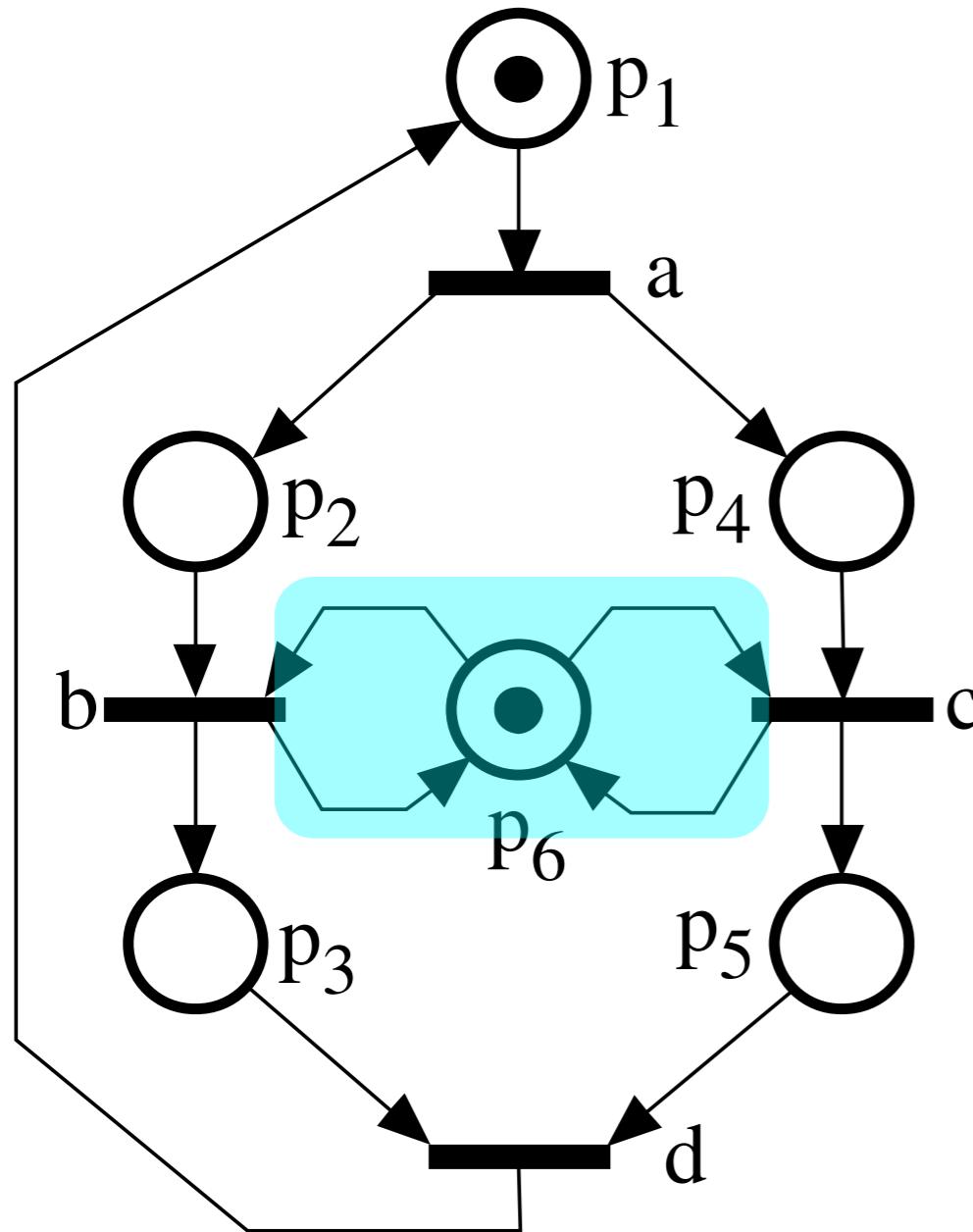
FGI 2

Daniel Moldt

Petrinetzprozesse (teilweise Wdh.)

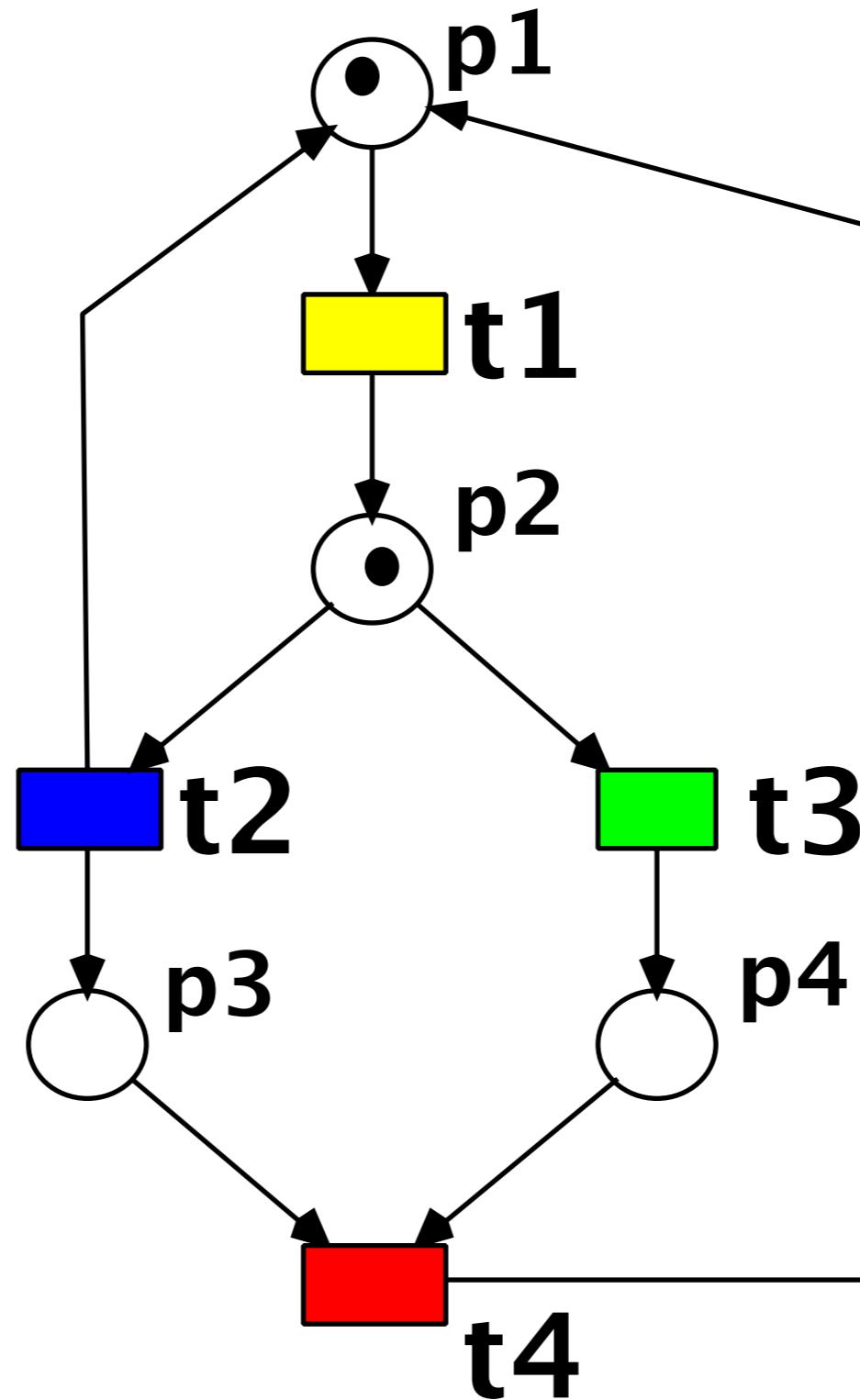
Erreichbarkeitsgraph

Der Erreichbarkeitsgraph beschreibt nur eine **Folgensemantik**:

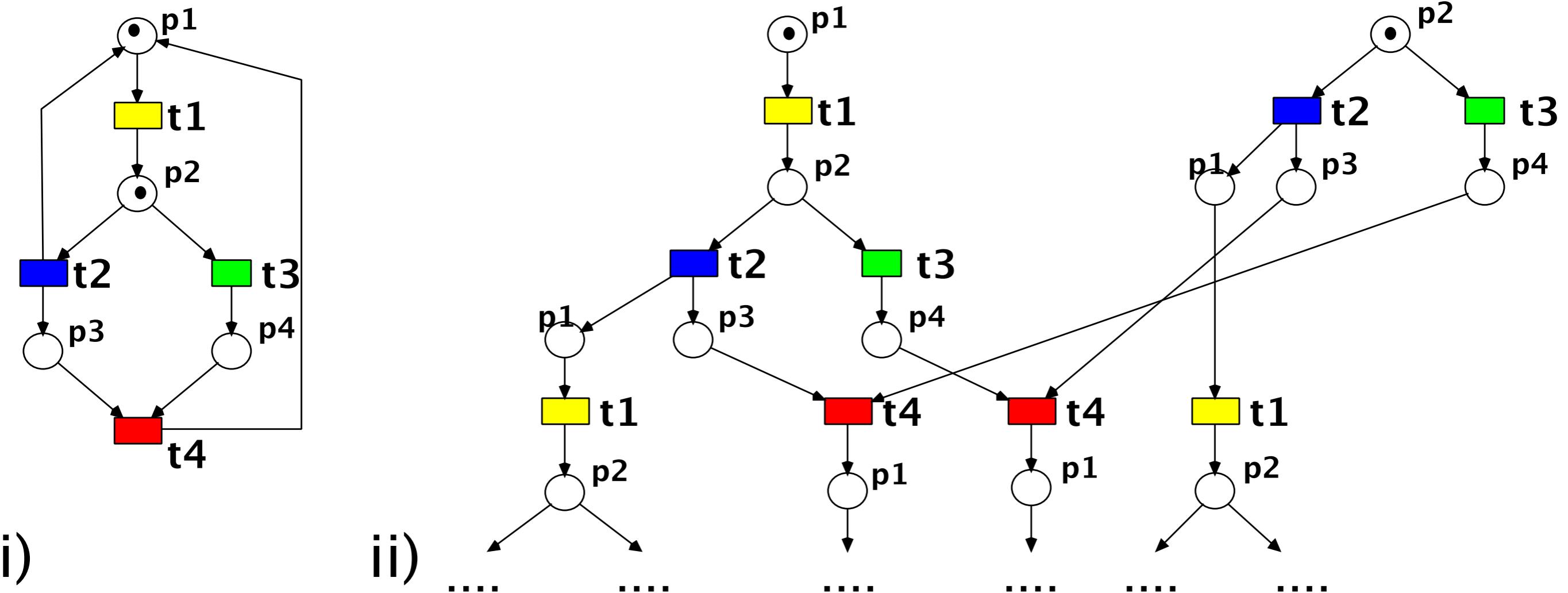


Also: Nebenläufigkeit von b und c wird **nicht** adäquat dargestellt!

Unfoldings von Netzen



Unfoldings von Netzen



Ereignisnetze, Kausalnetze

Die Prozesse eines Netzes sind Kausalnetze. Die abgeschwächte Form von Kausalnetzen sind Ereignisnetze (engl. „occurrence nets“), die – im Gegensatz zu Kausalnetzen – noch vorwärtsverzweigende Stellen erlauben, beispielsweise um Systemkonflikte in Prozessen (engl. „branching processes“) darzustellen [Eng91].

Definition: Ein Petrinetz $N = (B, E, F)$ heißt **Ereignisnetz**, gdw. der transitive Abschluss F^+ azyklisch ist und für alle $b \in B$ auch $|\bullet b| \leq 1$ gilt. Gilt zusätzlich noch $|b^\bullet| \leq 1$, so ist N ein **Kausalnetz**.

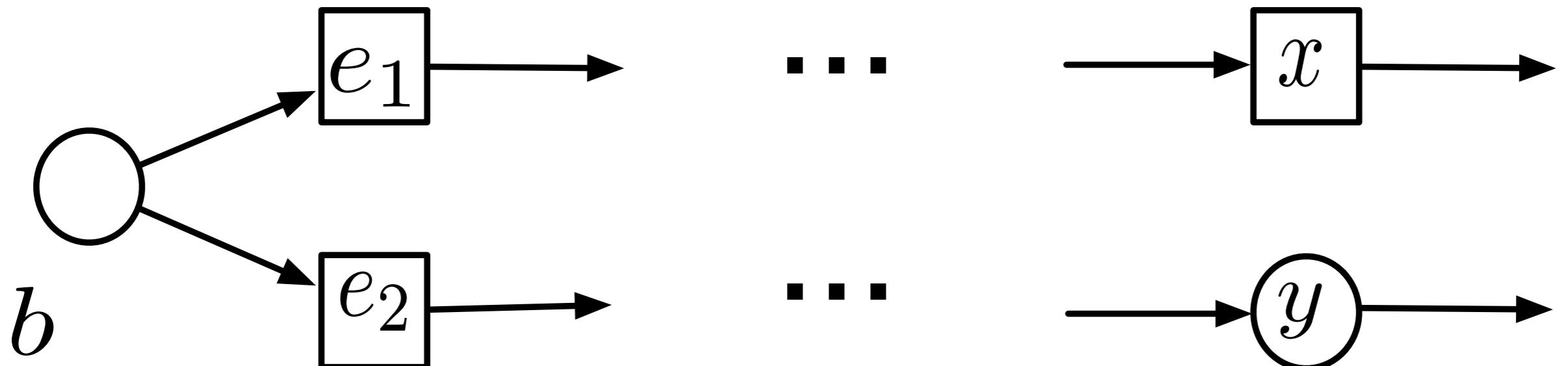
Für ein Ereignisnetz $N = (B, E, F)$ definiere die Ordnungen $<, \leq \subseteq (B \cup E)^2$ durch $< := F^+$ und $\leq := F^*$. Die Menge der Vorgänger eines Elements $y \in B \cup E$ ist $\downarrow y := (- < y) := \{x \mid x < y\}$. Das Ereignisnetz N heißt *vorgänger-endlich*, wenn für jedes $b \in B$ die Vorgängermenge $(- < b)$ endlich ist.

Konfliktrelation

Da ein Ereignisnetz vorwärtsverzweigt, können Netzelemente in Konflikt stehen, nämlich dann, wenn eine Stelle $b \in B$ existiert, von der zwei Konflikttereignisse $e_1, e_2 \in b^\bullet$ ausgehen. Dies wird durch die symmetrische **Konfliktrelation** $\# \subseteq (B \cup E)^2$ beschrieben:

$$x \# y \iff \exists b \in B : \exists e_1, e_2 \in b^\bullet : e_1 \neq e_2 \wedge e_1 \leq x \wedge e_2 \leq y$$

Kausalnetze sind demnach immer konfliktfrei: $\# = \emptyset$.



Prozesse

Definition: Sei $N = (P, T, F, W, \mathbf{m}_0)$ ein P/T-Netz, $R = (B, E, \lessdot)$ ein vorgängerendliches Ereignisnetz und ϕ ist ein Abbildungspaar aus $\phi_P : B \rightarrow P$ und $\phi_T : E \rightarrow T$, dann ist (R, ϕ) ein **Verzweigungsprozess** des Netzes N , falls die folgenden Eigenschaften gelten:

1. Zu jedem Knoten in R ist die Anzahl der Vorgänger endlich.
2. Die Stellen ${}^\circ R$ beschreiben \mathbf{m}_0 : $\phi_P^\oplus({}^\circ R) = \mathbf{m}_0$.
3. Die Prozessabbildung ϕ ist verträglich mit der Kantengewichtung:

$$\phi_P^\oplus(\bullet e) = \partial_0(\phi_T(e)) \quad \text{und} \quad \phi_P^\oplus(e^\bullet) = \partial_1(\phi_T(e))$$

Ist R zudem noch ein Kausalnetz, so heißt (R, ϕ) ein *Prozess* des Netzes N .

Die Menge aller Verzweigungsprozesse von N wird mit **BranProc(N)** bezeichnet,
Die Menge aller Prozesse mit **Proc(N)**.

Konstruktion: “Anhängen”

Die Konstruktion erzeugt aus einem bestehenden Prozess induktiv einen weiteren, indem zum alten Prozessnetz – im Einklang mit dem Netz N – weitere Transitionen und Stellen hinzugefügt werden.

Definition: Sei $N = (P, T, F, W, \mathbf{m}_0)$ ein P/T-Netz.

1. Sei C eine Menge mit $\phi(C) = \mathbf{m}_0$, dann ist $((C, \emptyset, \emptyset), \phi)$ ein Prozess.
2. Sei $(R, \phi) = ((B, E, \lessdot), \phi)$ ein Prozess und $t \in T$ eine Transition, für die die Menge $A_t \subseteq R^\circ$ jede Eingangsstelle $p \in {}^\bullet t$ mindestens $W(p, t)$ -mal enthält: $|\phi^{-1}(p) \cap A_t| = W(p, t)$. Mit

$$\begin{aligned} B' &= B \uplus B'', B'' = \{b''_{p,i} \mid p \in t^\bullet, 1 \leq i \leq W(t, p)\} \\ E' &= E \uplus \{e\} \\ \lessdot' &= \lessdot \cup \{(b, e) \mid b \in \phi^{-1}(p) \cap A_t\} \cup \{(e, b) \mid b \in B''\} \\ \phi' &= \phi \cup \{(e, t)\} \cup \{(b''_{p,i}, p) \mid b''_{p,i} \in B''\} \end{aligned}$$

ergibt sich dann auch $((B', E', \lessdot'), \phi')$ als Prozess von N .

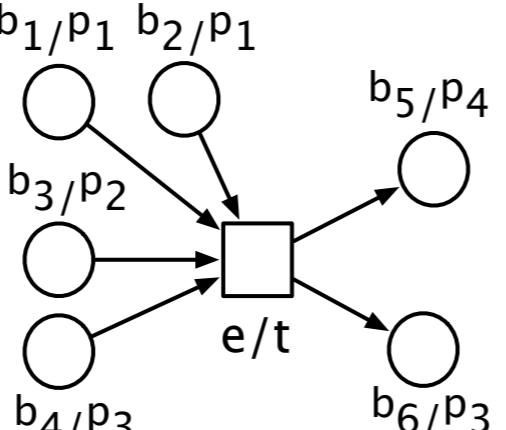
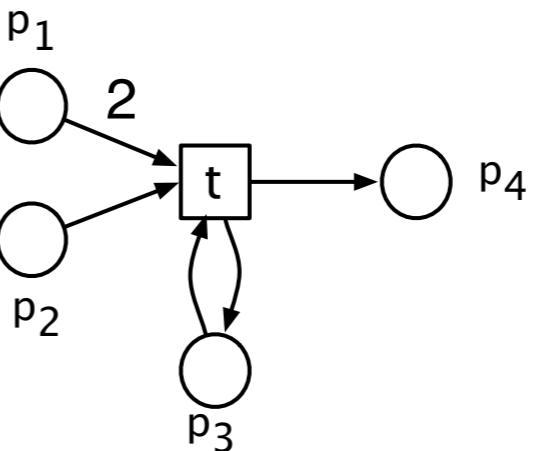
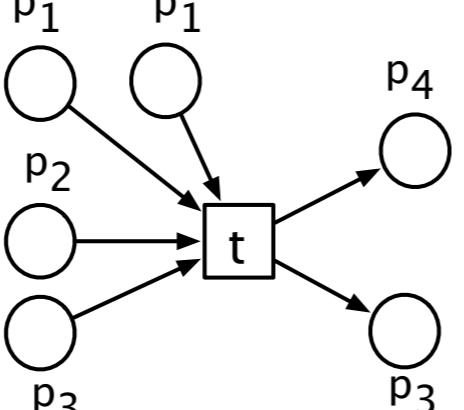
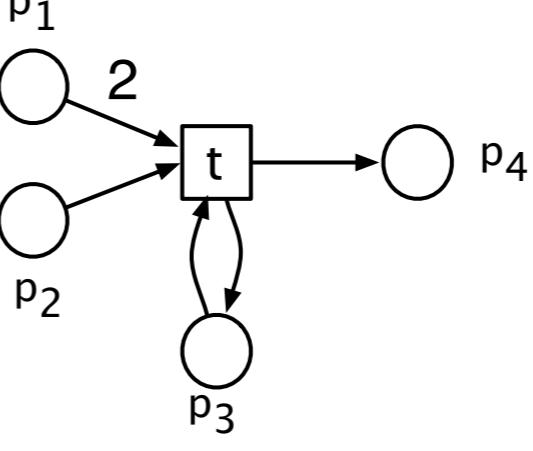
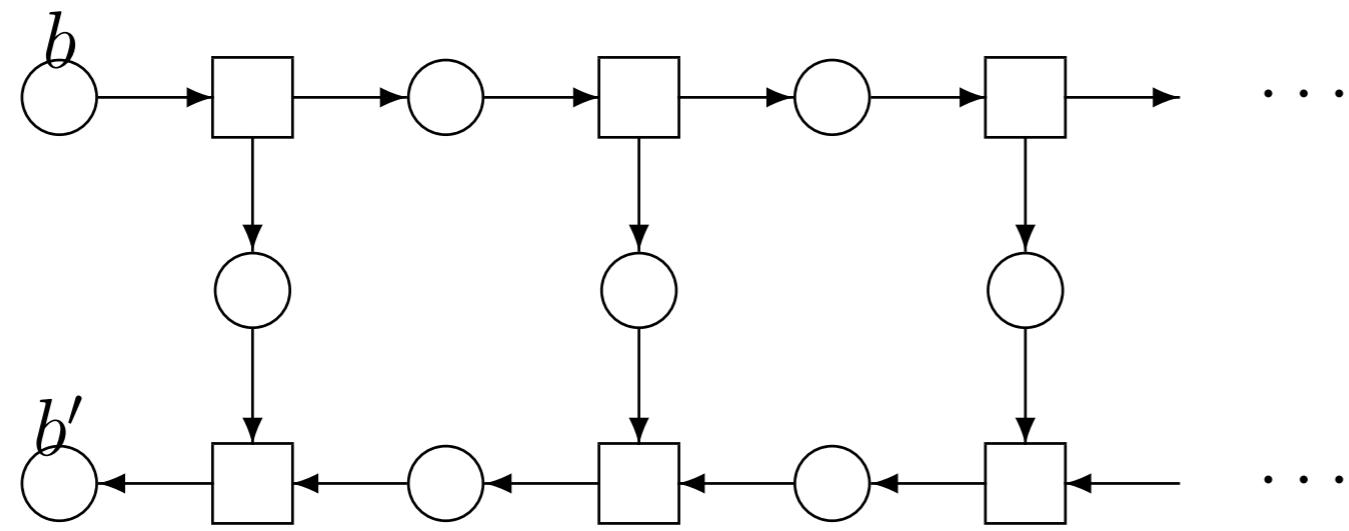
(a)	$b_1/\phi(b_1)=p$ $b_2/\phi(b_2)=p$	repräsentiert	
(b)	b_1/p b_2/p	repräsentiert	
(c)	p p	repräsentiert	
(d)	 		
(e)	 		

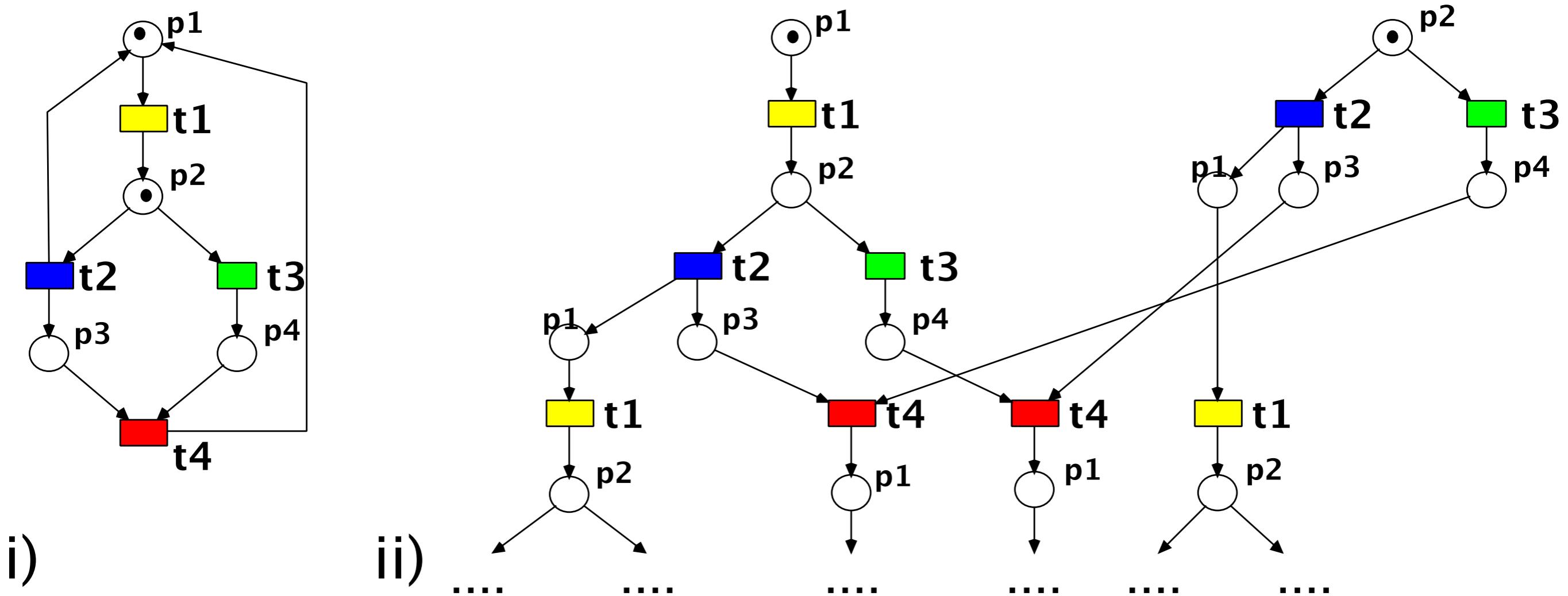
Abbildung 6.22: Aktion einer Transition



Abbildung

Ein unendlicher, nicht vorgänger-endlicher Prozess

Bsp.: Verzweigungsprozess

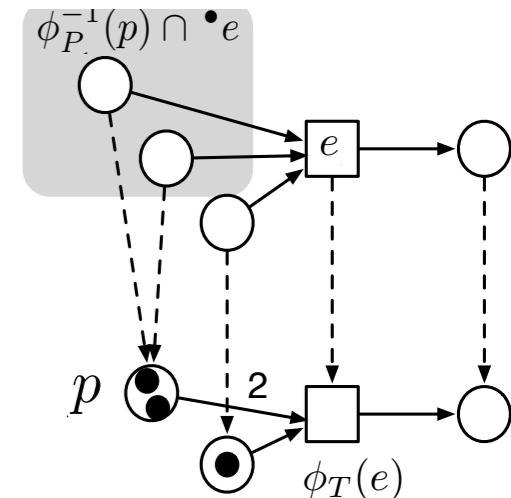


Definition 6.31 Sei $N = (P, T, F, W, \mathbf{m}_0)$ ein P/T-Netz, $R = (B, E, F_R)$ ein vorgängerendliches Ereignisnetz und ϕ ist ein Abbildungspaar aus $\phi_P : B \rightarrow P$ und $\phi_T : E \rightarrow T$, dann ist (R, ϕ) ein Verzweigungsprozess des Netzes N , falls die folgenden Eigenschaften gelten:

1. Die Struktur von R „passt“ zu N : $(x, y) \in F_R \Rightarrow (\phi(x), \phi(y)) \in F$
2. Die Stellen ${}^\circ R$ beschreiben \mathbf{m}_0 : $\forall p \in P : \mathbf{m}_0(p) = |\phi^{-1}(p) \cap {}^\circ R|$
3. Die Prozessabbildung ϕ ist verträglich mit der Kantengewichtung:

$$\forall e \in E \quad \forall p \in {}^\bullet \phi_T(e) : W(p, \phi_T(e)) = |\phi_P^{-1}(p) \cap {}^\bullet e| \quad \text{und}$$

$$\forall e \in E \quad \forall p \in \phi_T(e)^\bullet : W(\phi_T(e), p) = |\phi_P^{-1}(p) \cap e^\bullet|$$



Ist R zudem noch ein Kausalnetz, so heißt (R, ϕ) ein Prozess des Netzes N .

Die Menge aller Verzweigungsprozesse von N wird mit BranProc(N) bezeichnet, die Menge aller Prozesse mit Proc(N).

Definition 6.32 Sei $N = (P, T, F, W, \mathbf{m}_0)$ ein P/T -Netz.

1. Sei C eine Menge mit $\phi(c) \in P$ derart, dass $\phi^{-1}(p) = \mathbf{m}_0(p)$ für alle $p \in P$.
((C, \emptyset, \emptyset), ϕ) ist ein **Prozess (Prozessanfang)**. Für das Netz in Abbildung 6.24 (a) ist dies z.B. die Menge c_1, c_2, c_3, c_4 mit $\phi(c_1) = p_1, \phi(c_2) = p_1, \phi(c_3) = p_1, \phi(c_4) = p_4$ in Abbildung 6.24 (b). Streicht man hier die anderen Transitionen und Plätze, so erhält man den Anfangsprozess, der der Anfangsmarkierung entspricht.
2. Sei $(R, \phi) = ((B, E, F_K), \phi)$ ein (schon konstruierter) Prozess und $t \in T$ eine Transition. Sein **maximaler Schnitt R°** enthalte für jeden Eingangsplatz $p \in {}^*t$ mindestens $W(p, t)$ Elemente b_i mit $\phi(b_i) = p$. Für den Prozess in Abbildung 6.24 (b) und die Transition c von (a) ist dies der Fall: R° enthält 2 Elemente b_i mit $\phi(b_i) = p_1$ und ein Element b_i mit $\phi(b_i) = p_4$. Es werden (außer dem Anfangsprozess) hier nur noch die Etikette angeben.
3. Der Prozess wird nun verlängert, indem eine neue Transition e mit $\phi(e) = t$ eingefügt wird, die die unter 2. genannten Plätze als Eingangsplätze enthält. Als Ausgangsplätze werden für jedes $p \in t^\bullet$ eine Anzahl von $W(t, p)$ neuen Plätzen b_i mit $\phi(b_i) = p$ angefügt. Dies ist für das Beispiel in Abbildung 6.24 (c) ausgeführt (hier ist $t = c$ und $p = p_3$).

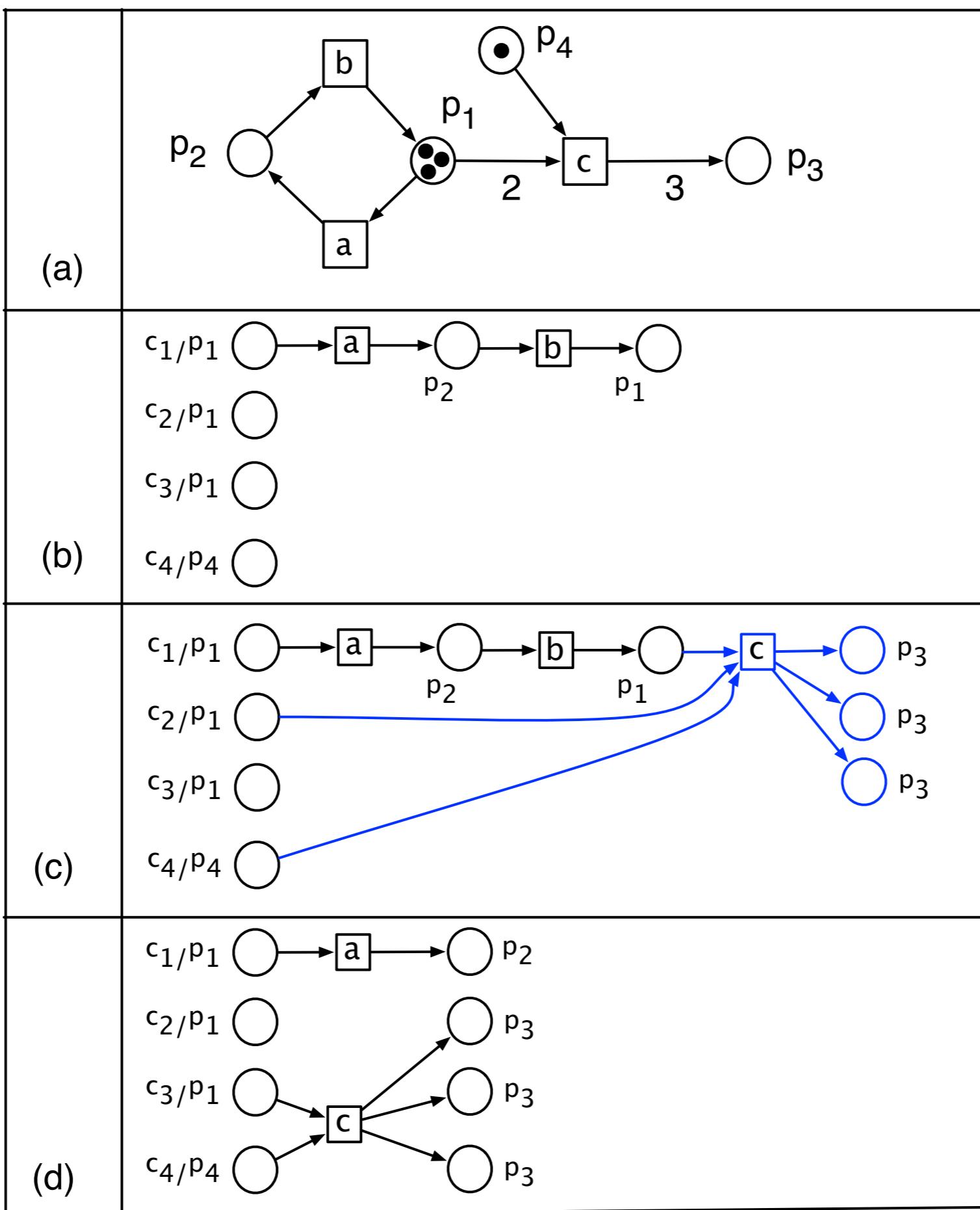
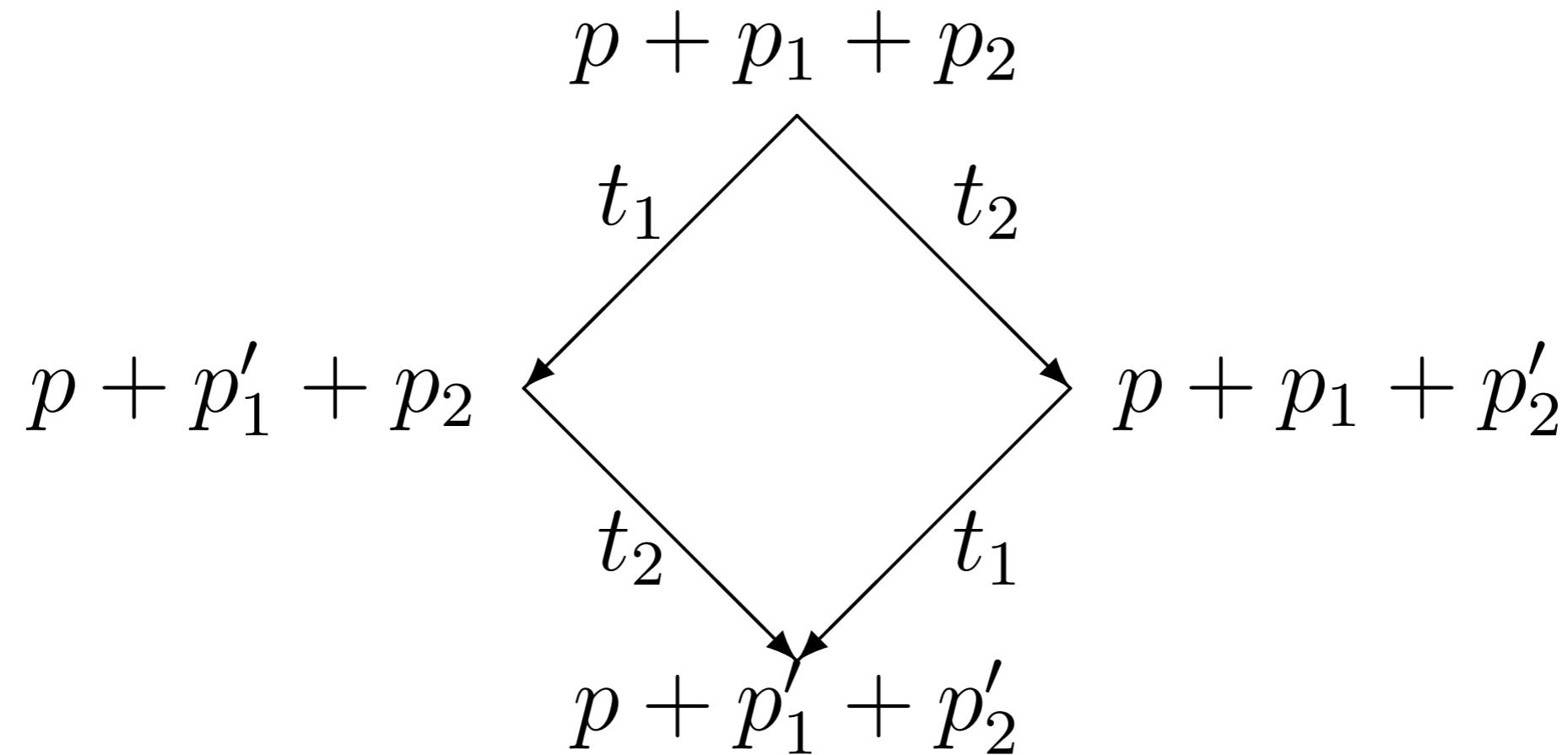
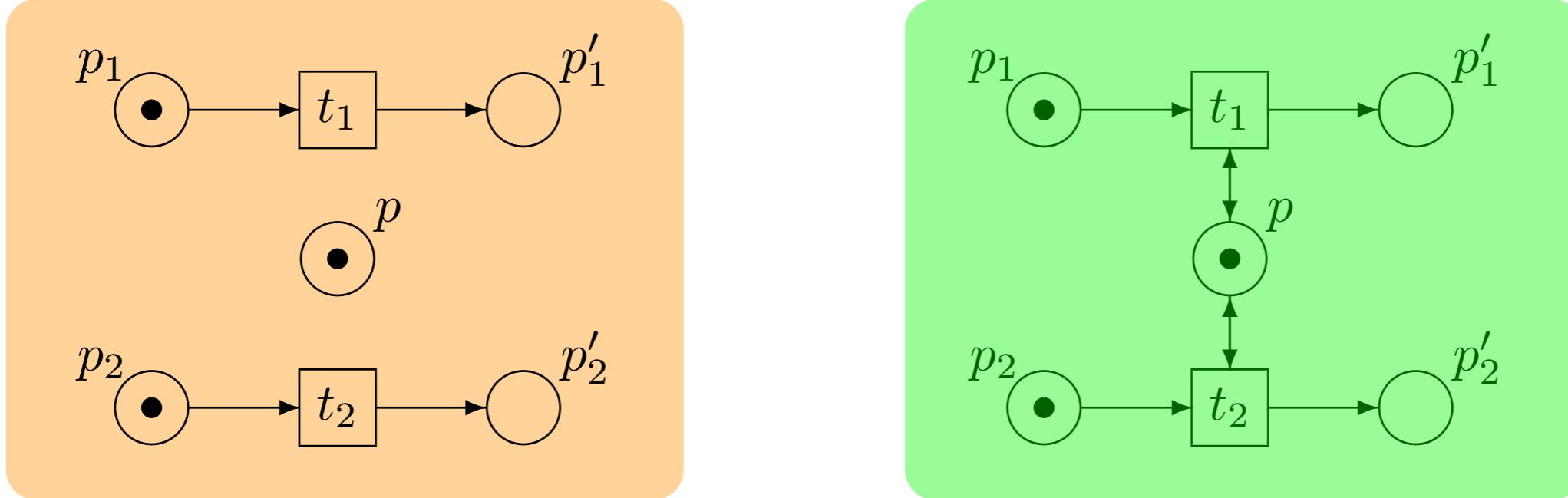
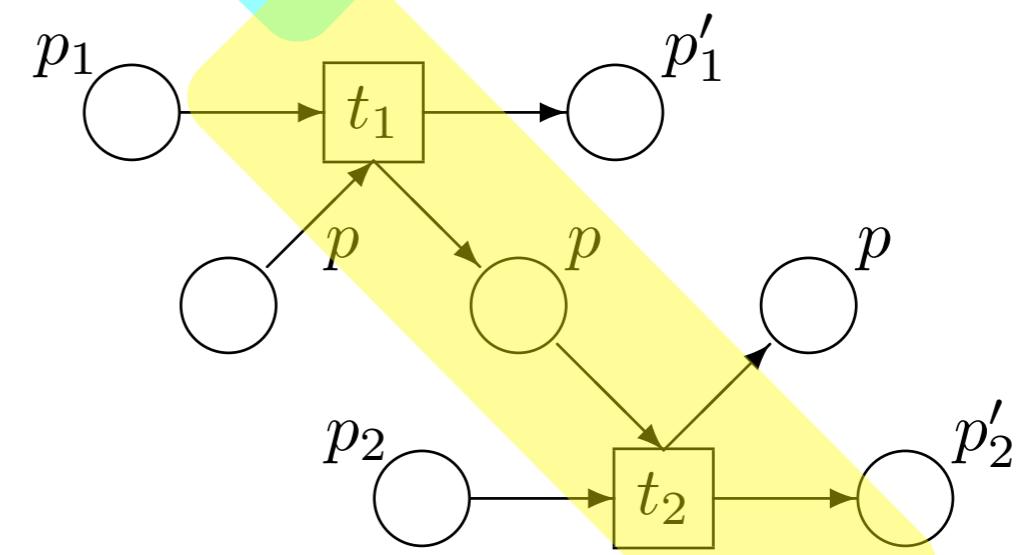
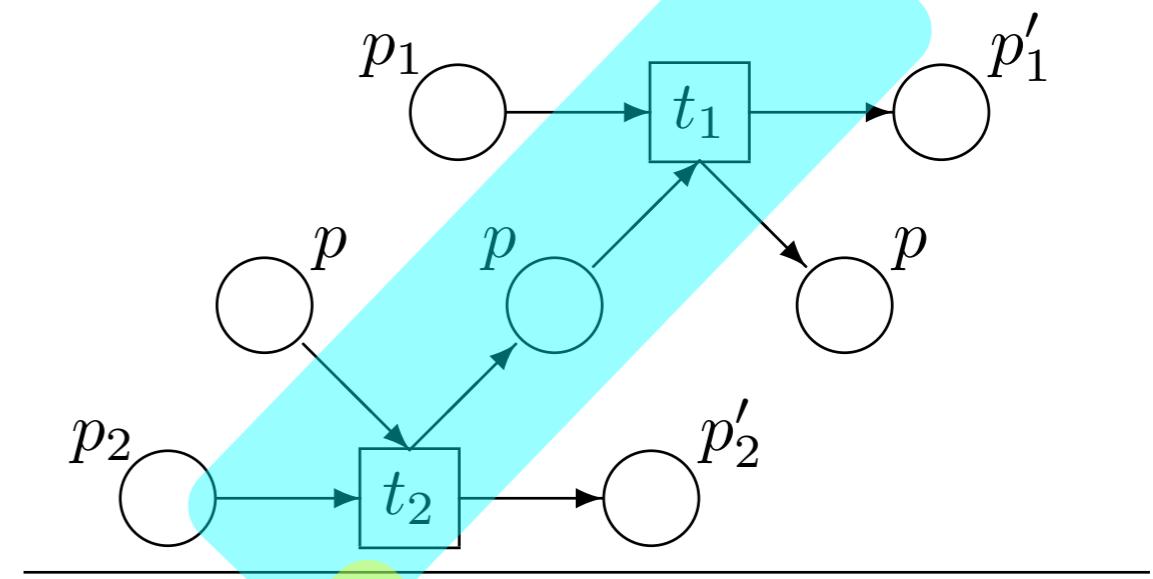
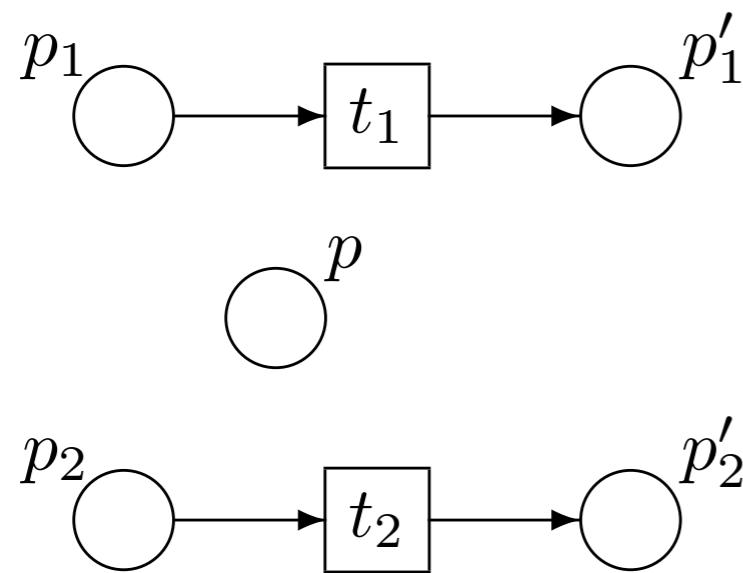
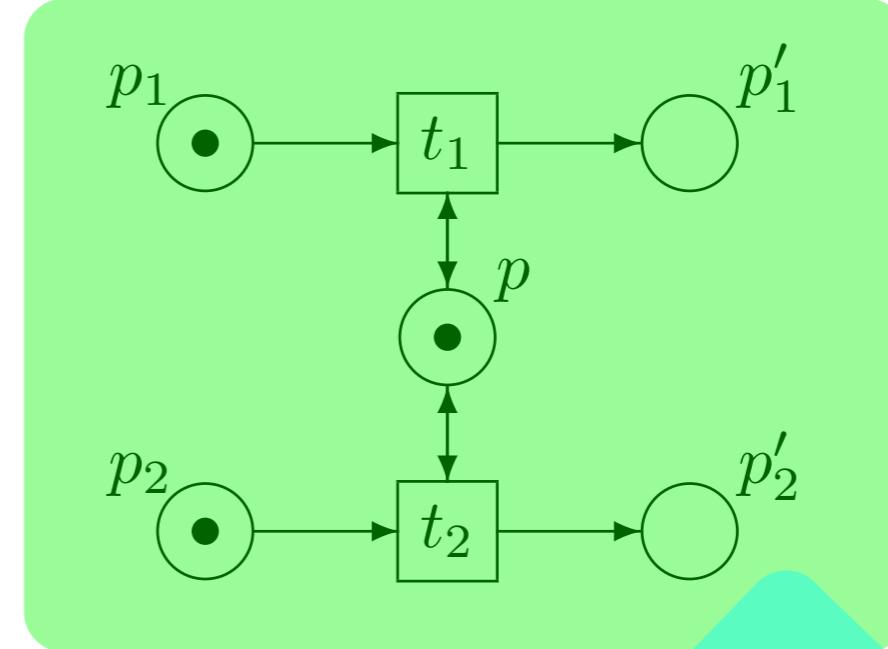
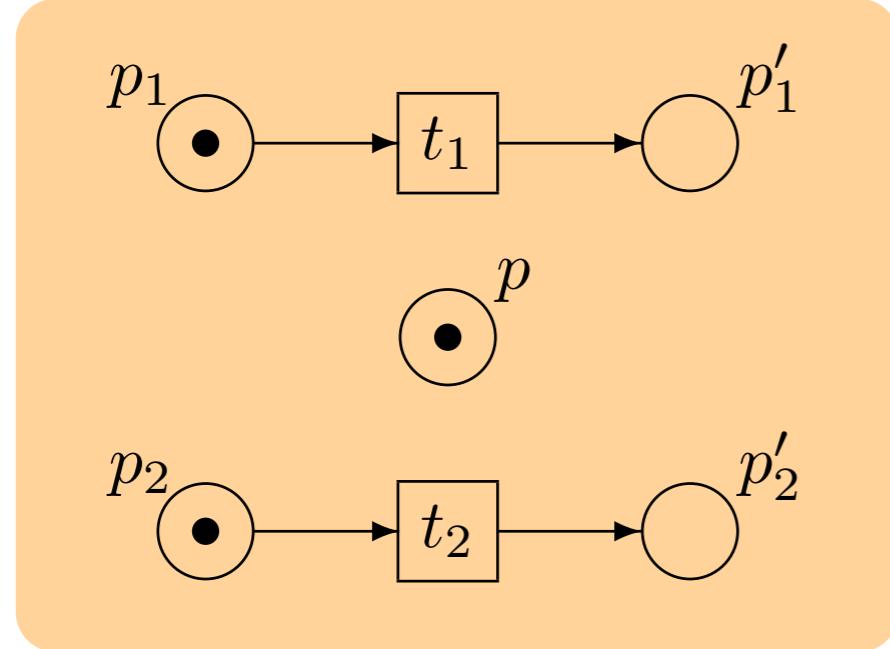


Abbildung 6.24: Zur 2. Prozessdefinition

Nebenläufigkeit vs. Permutation



Nebenläufigkeit vs. Permutation



Linien und Schnitte

Sei $R \subseteq A \times A$ eine symmetrische, reflexive Relation. $K \subseteq A$ heißt *Klique* bezüglich R , gdw. alle Elemente in Relation stehen, d.h. für alle $x, y \in K$ gilt $x R y$.

Eine maximale Klique heißt *Bezirk*.

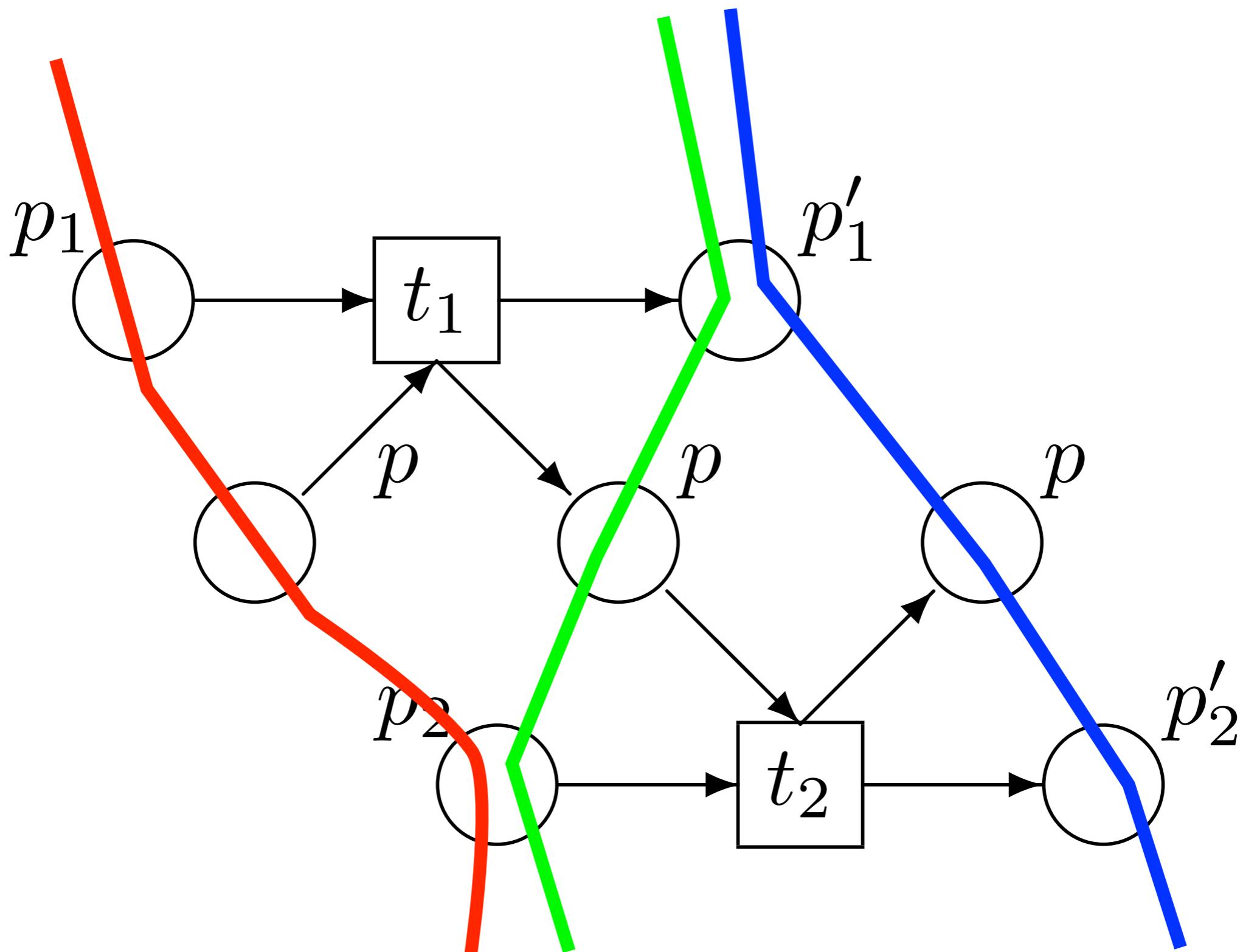
Die Menge aller Bezirke von R wird durch $\text{BEZ}(R)$ notiert.

Sei die Relation $<$ gegeben. Die symmetrischen und reflexiven Relationen **li** und **co** sind definiert als:

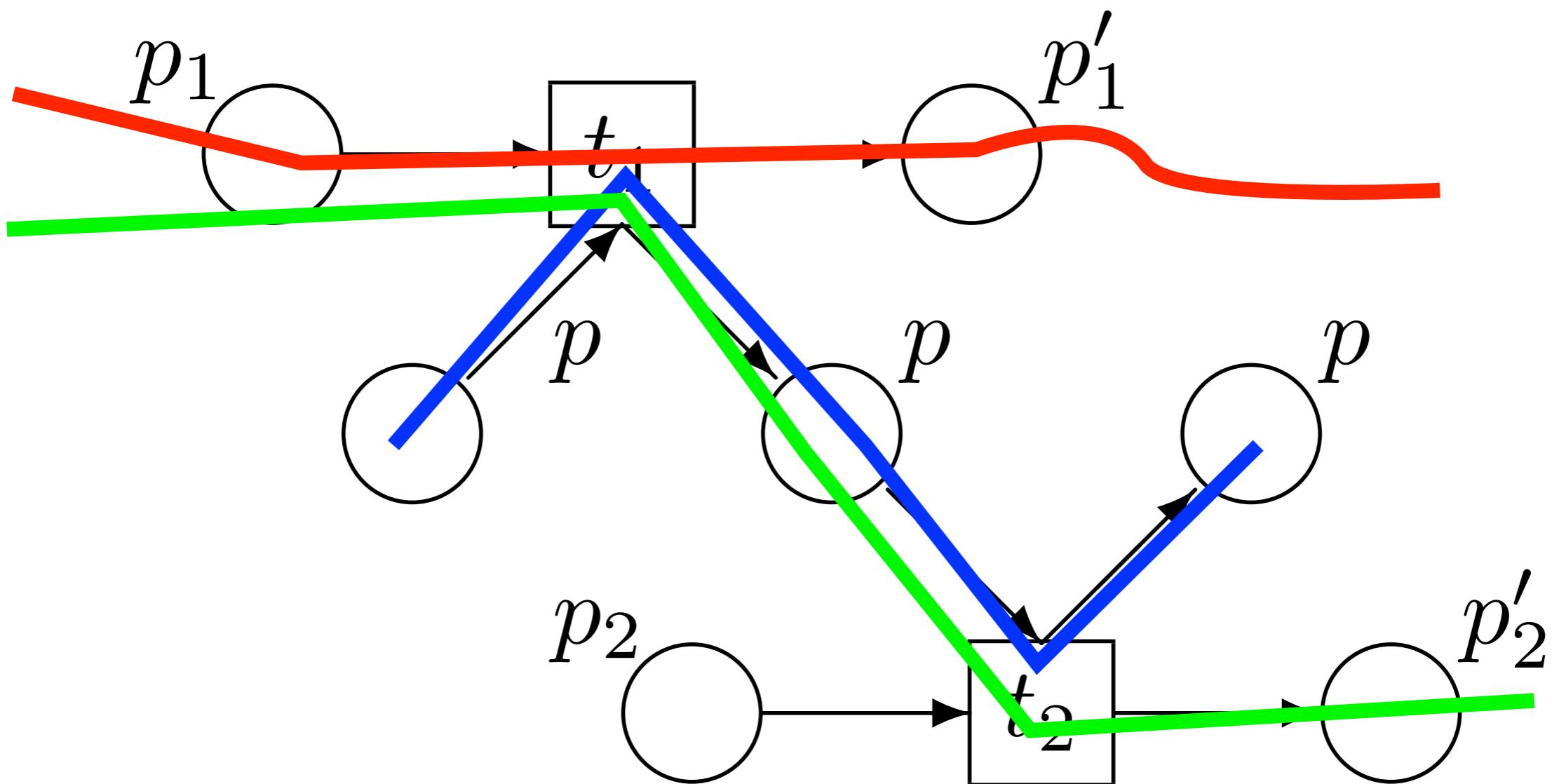
$$\mathbf{li} := < \cup <^{-1} \cup id_A \quad \text{und} \quad \mathbf{co} := \bar{\mathbf{li}} \cup id_A$$

Die Menge der Bezirke bezüglich **li** werden als *Linien*, die Bezirke bezüglich **co** als *Schnitte* bezeichnet.

Beispiel: Schnitte



Beispiel: Linien



Stellenschnitte

Die Menge der P -Schnitte eines Prozesses R wird im folgenden durch \mathcal{C}_R notiert. Insbesondere ergibt sich die Menge \mathcal{C}_R aller Stellen-Schnitte, kurz: P -Schnitte, als die Menge der möglichen Markierungen eines Prozesses:

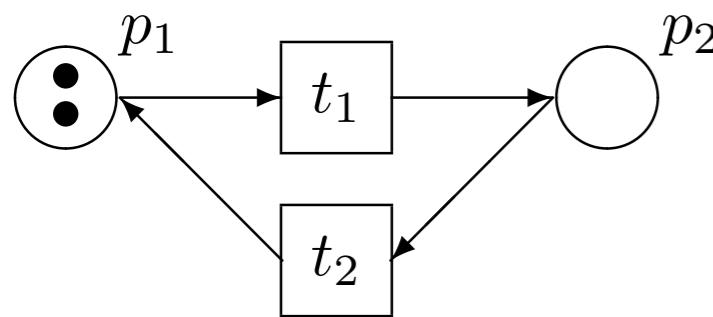
Theorem: Sei $(R, \phi) \in Proc(N)$ ein Prozess von N , dann ist jeder Stellen-schnitt C eine erreichbare Markierung:

$$C \in \mathcal{C}_R \Rightarrow \phi(C) \in RS(N)$$

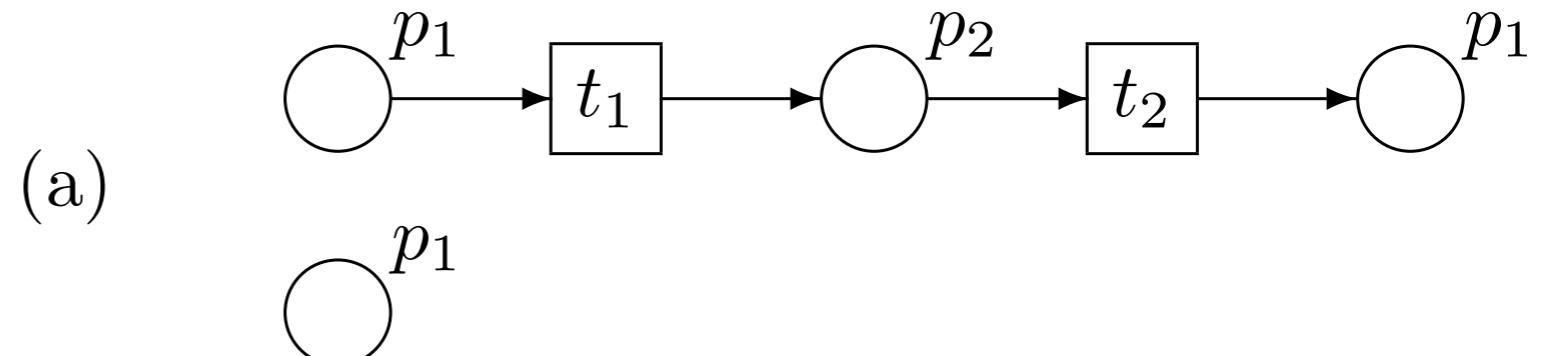
Die Transitions-Schnitte, kurz: T -Schnitte, sind die nebenläufigen Transi-tionen.

Sind black tokens unterscheidbar?

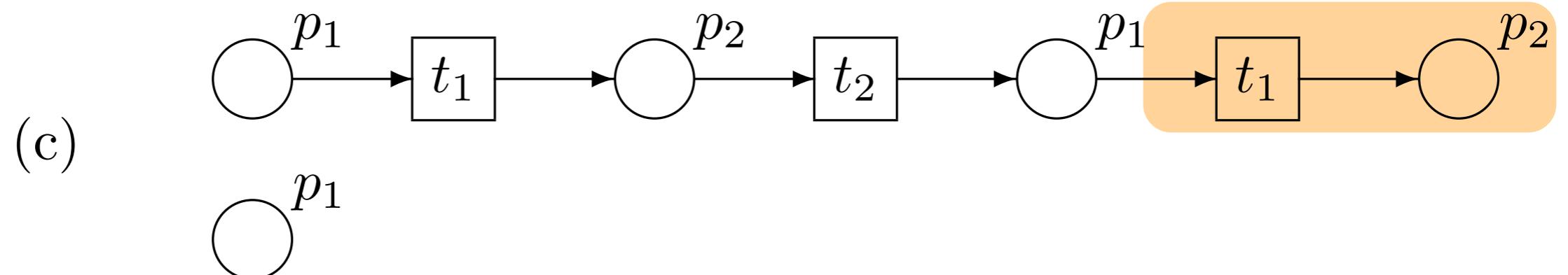
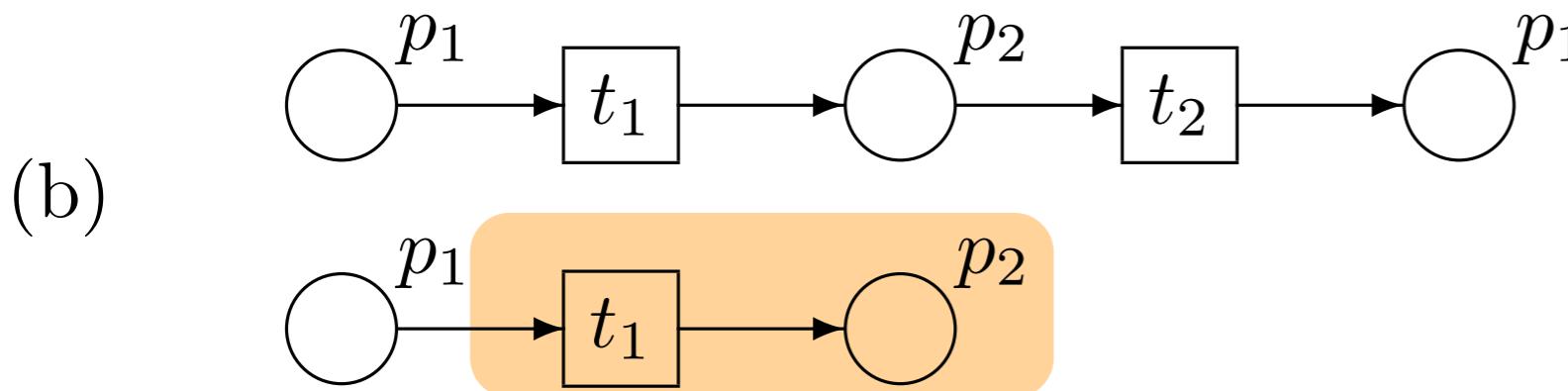
Petrinetz:



Schalthistorie bis jetzt:



Wir haben jetzt die Wahl zwischen **2 Varianten**, um t_1 zu schalten:



Fortschritt

Das Netz in Abbildung 1.20 erlaubt die Wiederholung der Aktionen a und b , d.h. $abababab\dots$ 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 die Fortschrittseigenschaft fordern.

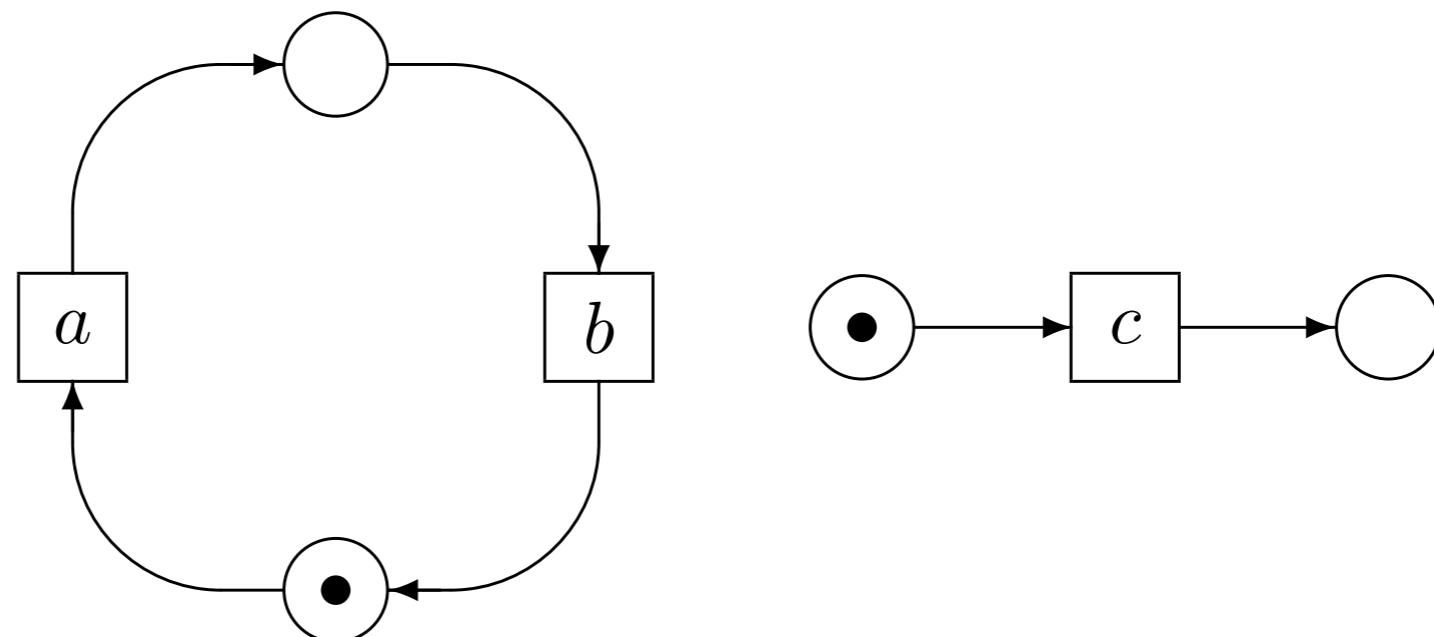


Abbildung 1.20: Zur Fortschrittseigenschaft

Fortschritt

Definition: Sei ein P/T-Netz N gegeben. Ein Prozess (R, ϕ) mit (B, E, \prec) 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:

$$\forall Q \subseteq R^\circ : \exists t \in T : \phi^\oplus(Q) = \partial_0(t) \Rightarrow t \notin T^p$$

Eine Transition $t \in T^p$ heißt Fortschritttransition. Positiv formuliert bedeutet Fortschritt, dass eine dauerhaft aktivierte Fortschritttransition $t \in T^p$ auch schalten muss.

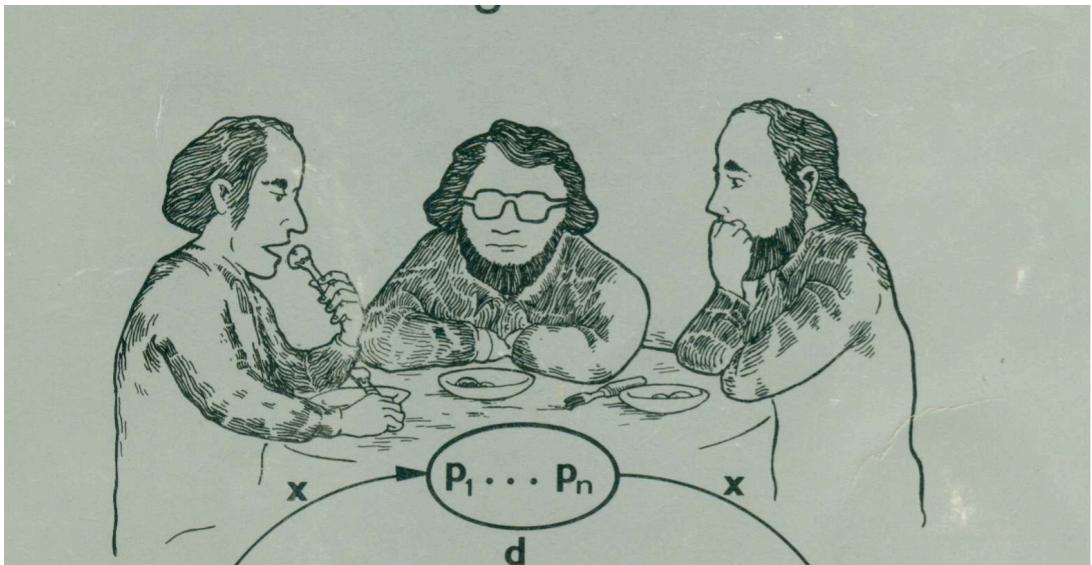
Eigenschaft: Wenn N ein lebendiges Netz ist und für alle Transitionen die Fortschrittseigenschaft gilt, dann sind alle Prozesse, welche die Fortschrittseigenschaft respektieren, unendlich.

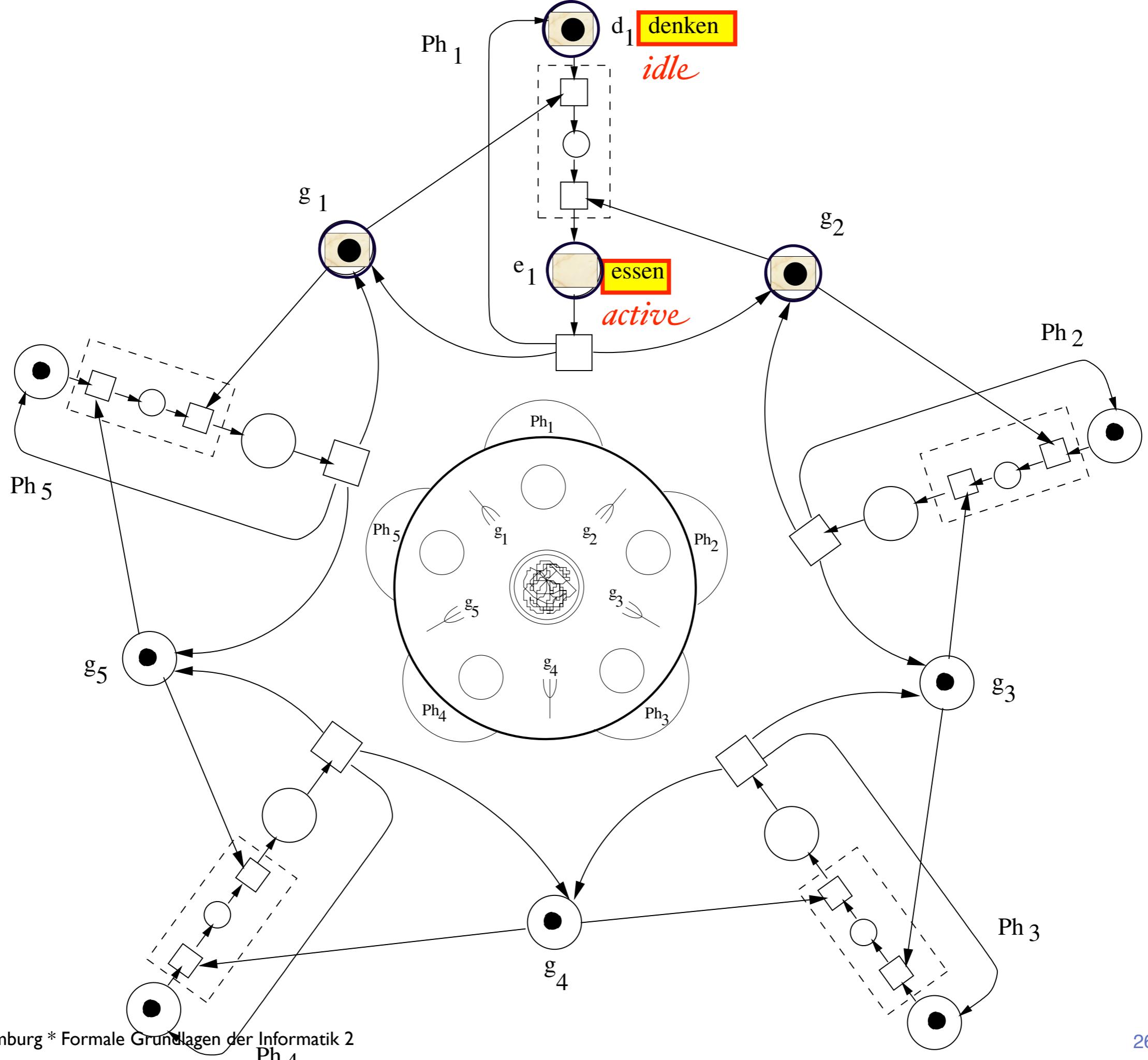
FGI 2

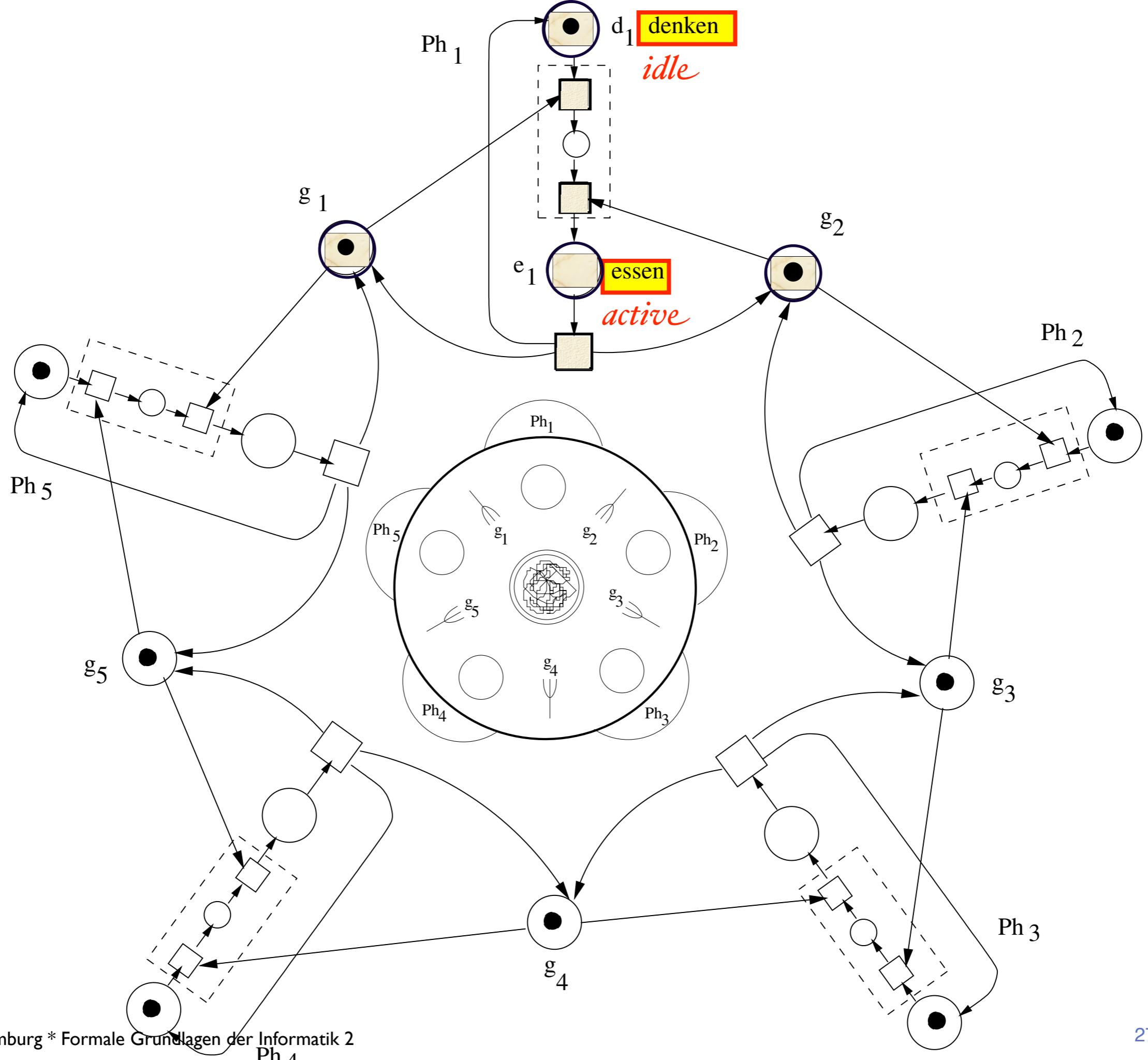
Daniel Moldt

Fairness

Fairness: 5 Philosophen





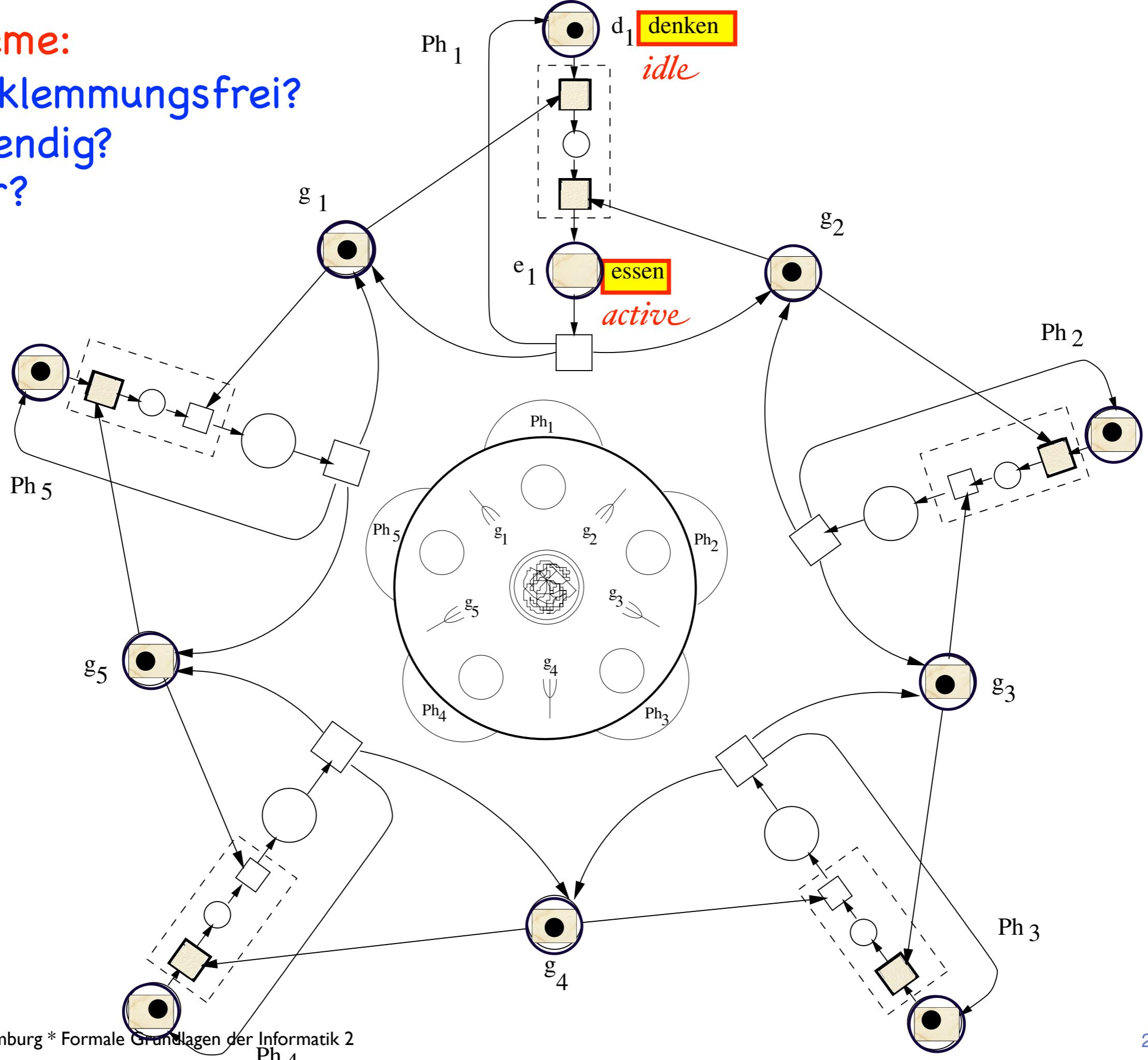


Probleme:

verklemmungsfrei?

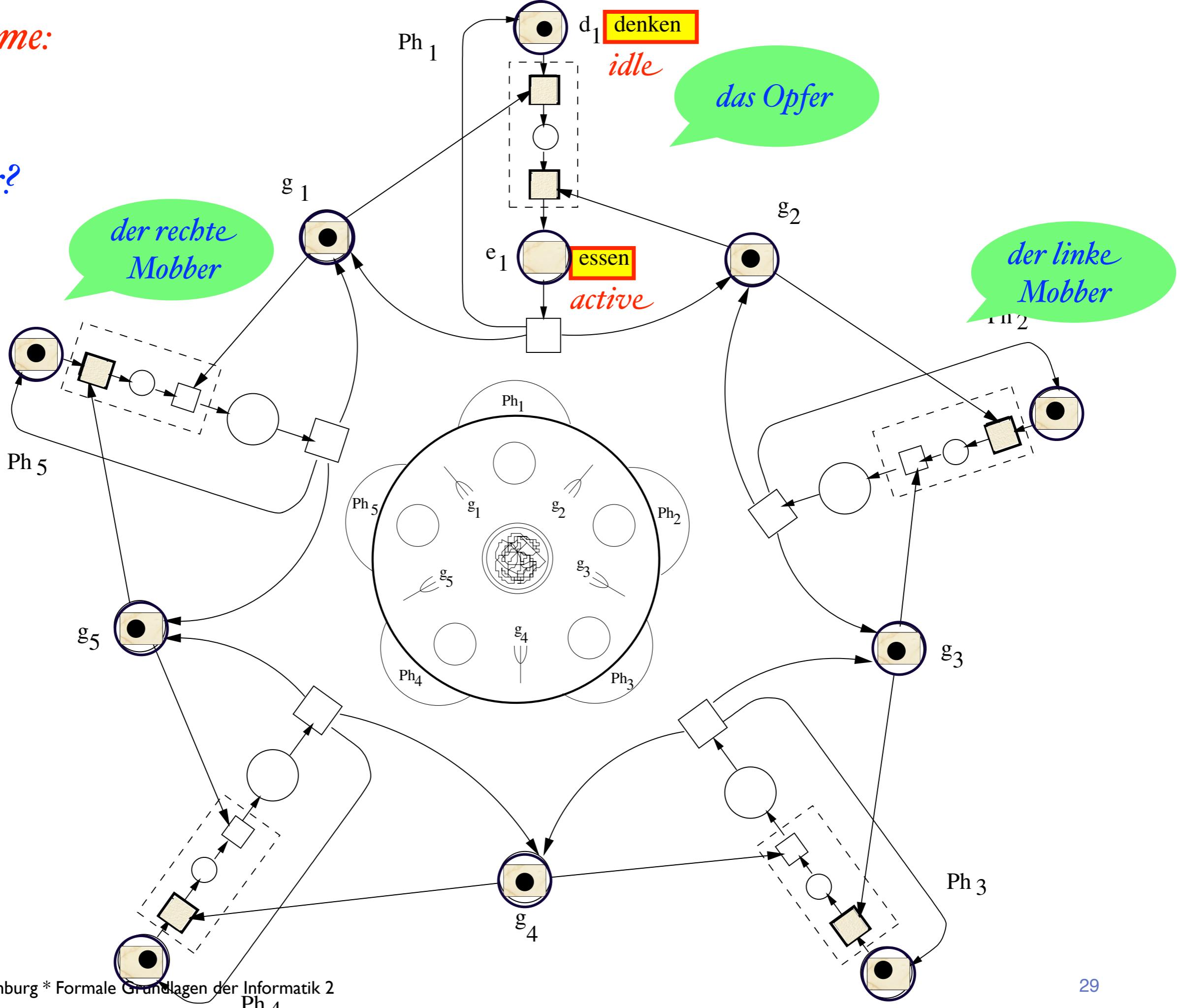
lebendig?

fair?



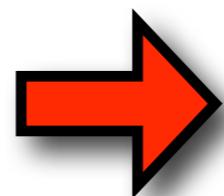
Probleme:

fair?



Fairness als globale Eigenschaft

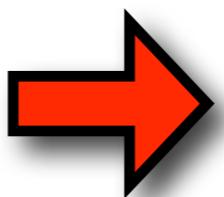
faire Schaltregel



faires Verhalten

?

lokales Verhalten



globales Verhalten

?

Faires Verhalten

Definition: Ein P/T -Netz $\mathcal{N} = \langle S, T, F, W, \mathbf{m}_0 \rangle$ 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 $F_\omega(\mathcal{N}) := \{w = a_1 a_2 a_3 \cdots \in T^\omega \mid \forall i \geq 1 : a_1 a_2 \cdots a_i \in FS(\mathcal{N})\}$.

Wir vergleichen die wichtigen Begriffe der Lebendigkeit und Fairness.

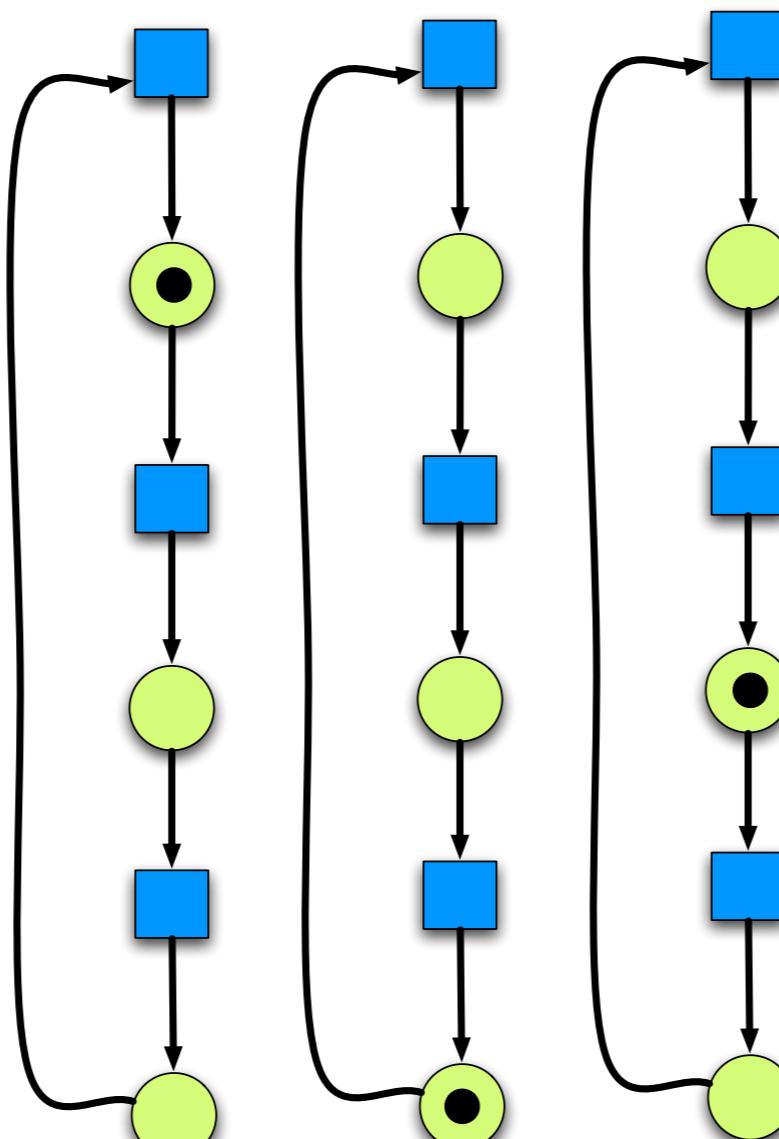
- a) Lebendigkeit bedeutet Freiheit von *unvermeidbaren* partiellen Verklemmungen.
- b) Fairness bedeutet Freiheit von *faktischen* partiellen Verklemmungen.

Verschleppungsfreie Schaltregel

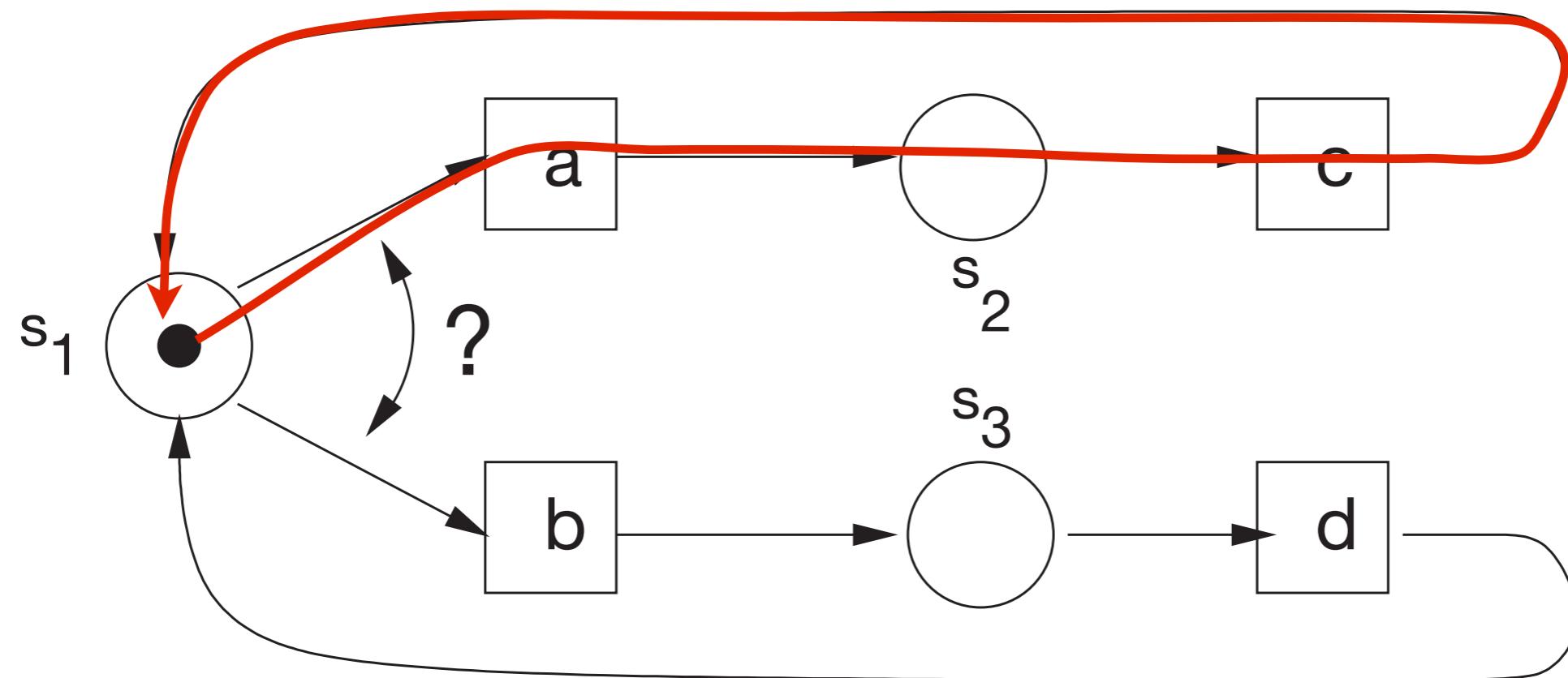
Definition 5.13 Es sei $\mathcal{N} = \langle S, T, F, W, \mathbf{m}_0 \rangle$ ein S/T -Netz.

a) \mathcal{N} schaltet produktiv oder verschleppungsfrei oder nach der verschleppungsfreien Schaltregel (*finite delay property*), wenn

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

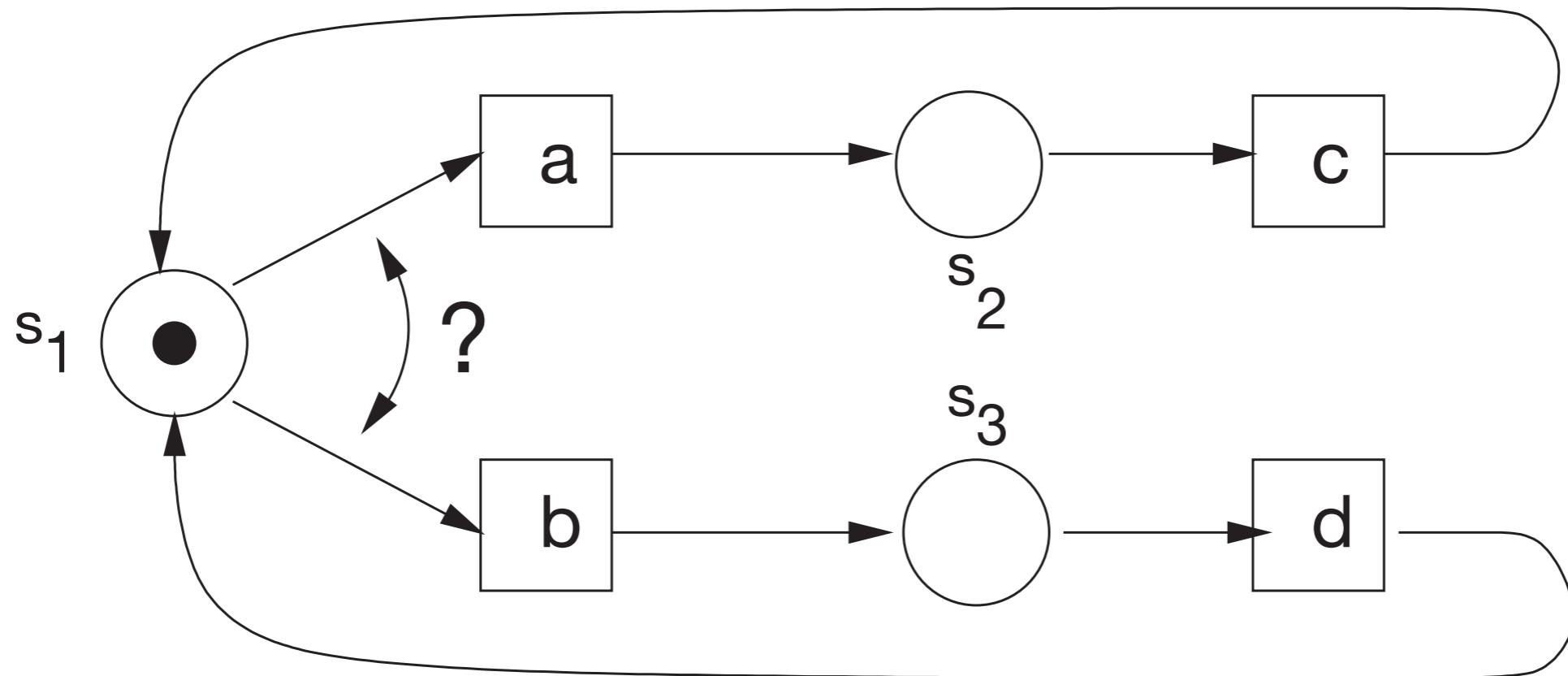


Verschleppungsfreie Schaltregel reicht nicht aus!



Das Netz von Abb.5.12 ist zwar lebendig, hat aber auch unter der verschleppungsfreien Schaltregel kein faires Verhalten ($ac\ ac\ \dots$ ist immer noch möglich). Das Netz verhält sich jedoch fair bei der fairen Schaltregel.

Faire Schaltregel



\mathcal{N} schaltet fair, oder nach der fairen Schaltregel (*fair firing rule*),
wenn

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

Definition 5.13 Es sei $\mathcal{N} = \langle S, T, F, W, \mathbf{m}_0 \rangle$ ein S/T -Netz.

a) \mathcal{N} schaltet produktiv oder verschleppungsfrei oder nach der verschleppungsfreien Schaltregel (*finite delay property*), wenn

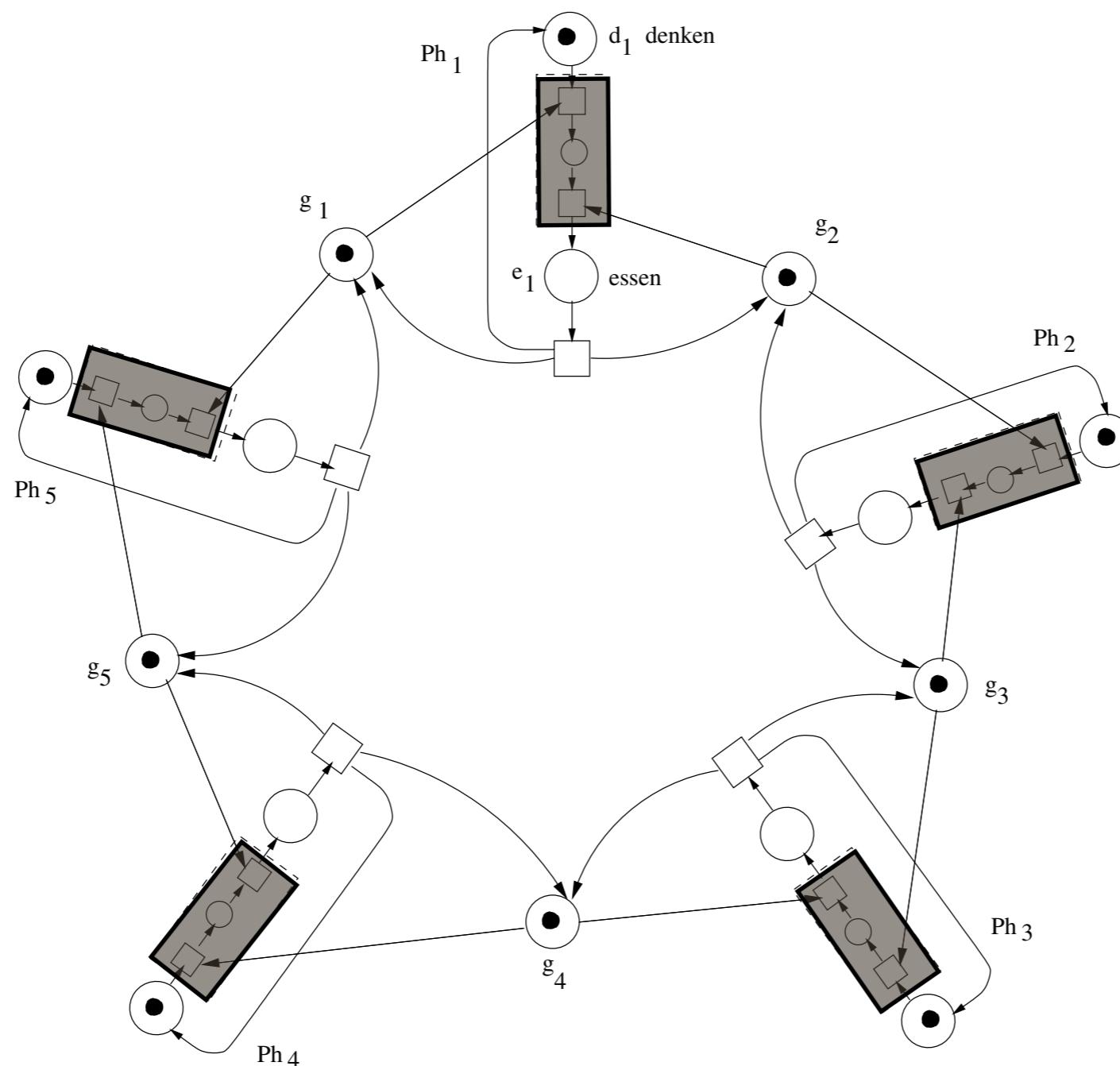
$$\forall t \in T \forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) \neg \exists w \in T^\omega : \mathbf{m} \xrightarrow{w} \wedge t \text{ ist bei } w \text{ permanent aktiviert} \wedge |w|_t = 0$$

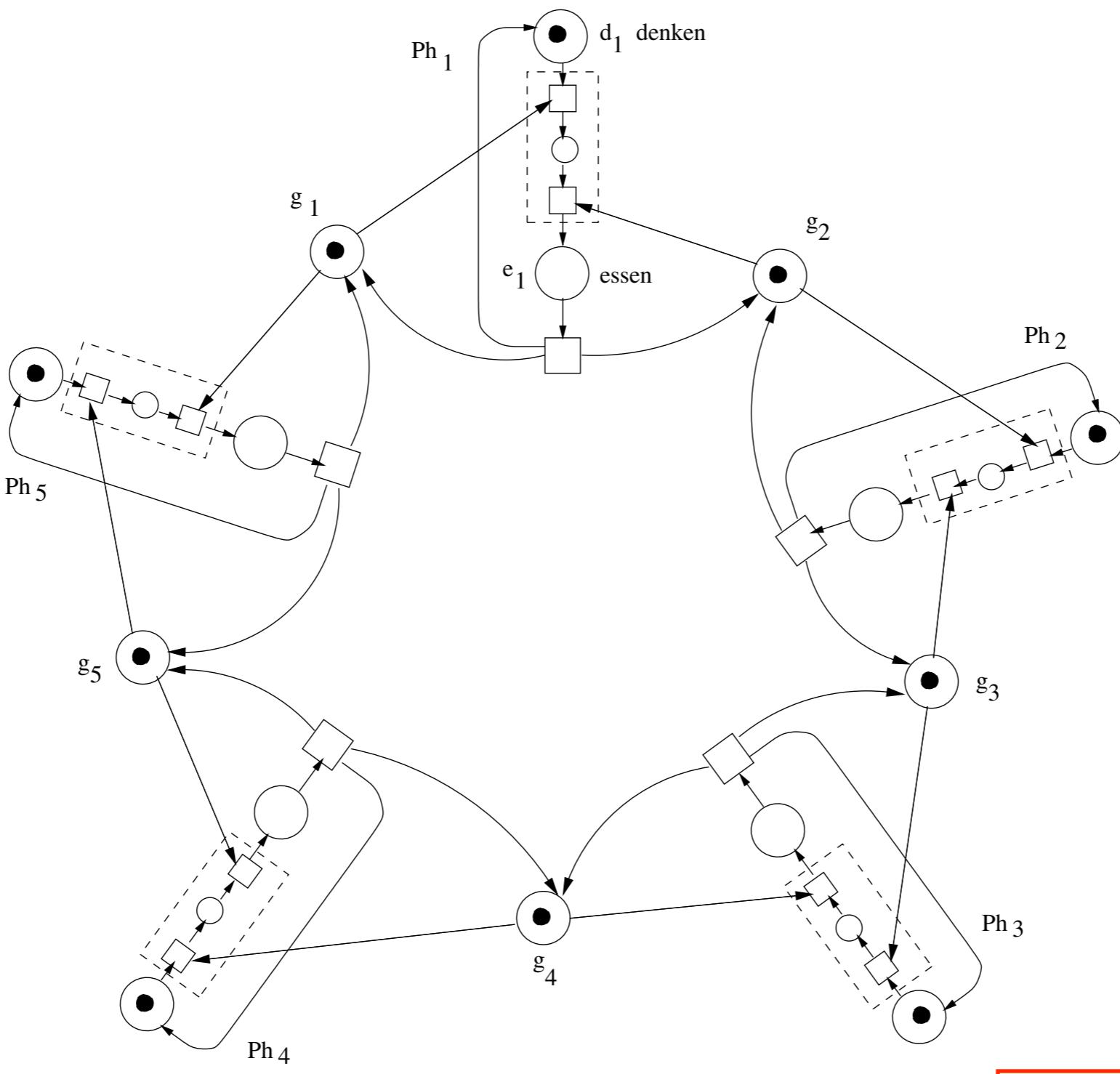
\mathcal{N} schaltet fair, oder nach der fairen Schaltregel (*fair firing rule*), wenn

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

Faire Schaltregel reicht nicht aus!

Das Netz von Abb.5.11 mit der vergröberten, und daher unteilbaren Transition ist **lebendig**. Es hat aber weder unter der verschlepungsfreien noch unter der fairen Schaltregel ein **faires Verhalten**.





- c) Das Netz von Abb.5.11 ohne Vergrößerung ist nicht lebendig. Unter der fairen Schaltregel hat es jedoch ein faires Verhalten. Verhindert man die Verklemmung durch andere Maßnahmen, z.B. indem man höchstens 4 Philosophen in den Essraum lässt (Abb. 6.22), dann ist das Netz lebendig und fair bei der fairen Schaltregel.

Exkurs: Starke und schwache Fairness

Fairness in LTL

unendlich oft φ :
ab irgendwann φ :

$\mathbf{G} \mathbf{F} \varphi$
 $\mathbf{F} \mathbf{G} \varphi$

(man schreibt auch \mathbf{F}^∞ für $\mathbf{G} \mathbf{F}$)
(man schreibt auch \mathbf{G}^∞ für $\mathbf{F} \mathbf{G}$)

Wichtig sind auch *Fairness-Aussagen*, die ausdrücken, dass jede Transition, die oft genug bereit ist, auch oft genug ausgeführt wird. Seien n Transitionen gegeben, dann drücken wir für $k = 1, \dots, n$ aus

enabled_k : Transition k ist bereit

executed_k : Transition k wird ausgeführt.

Unbedingte Fairness :

$$\bigwedge_{k=1, \dots, n} (\mathbf{G} \mathbf{F} \text{executed}_k)$$

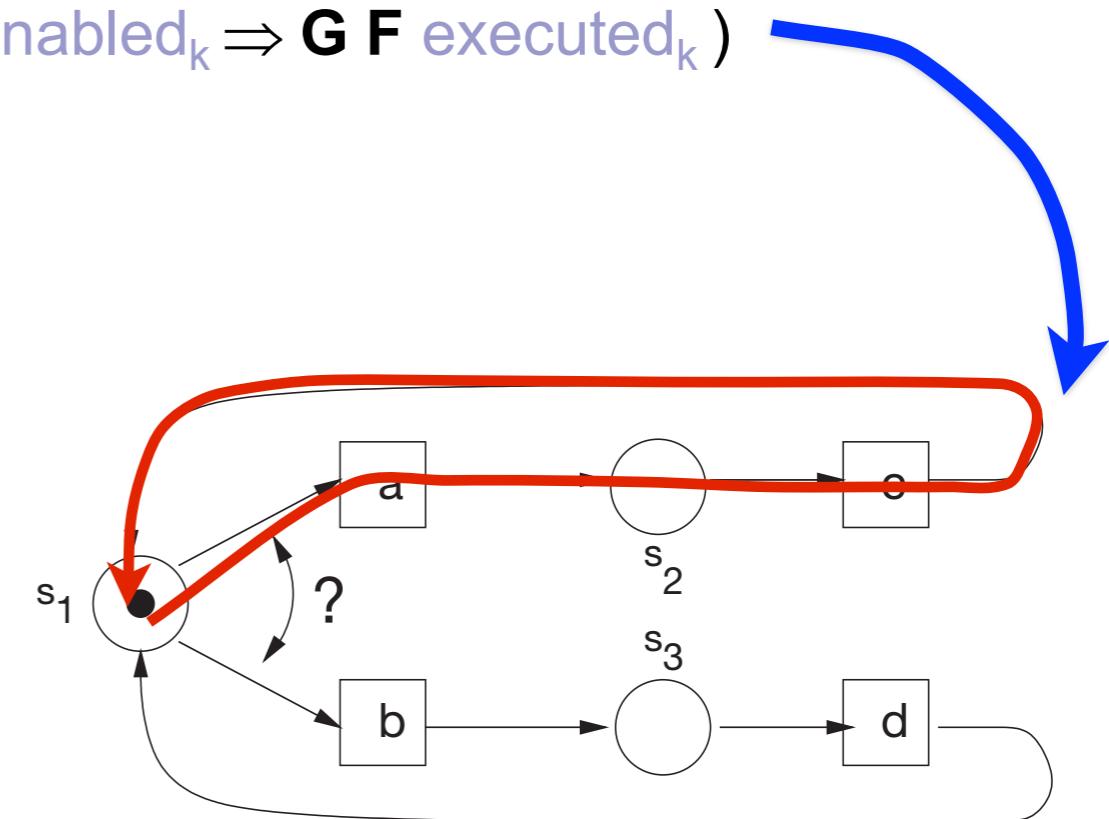
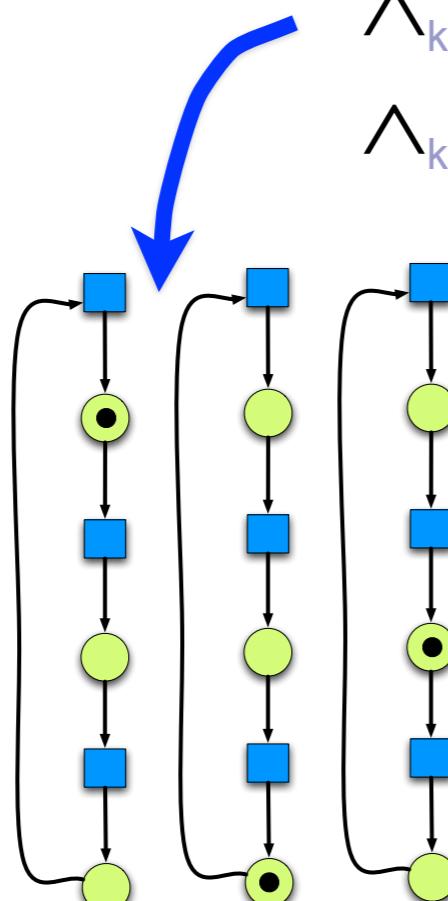
Alternierbit-Protokoll

Schwache Fairness :

$$\bigwedge_{k=1, \dots, n} (\mathbf{F} \mathbf{G} \text{enabled}_k \Rightarrow \mathbf{G} \mathbf{F} \text{executed}_k)$$

Starke Fairness :

$$\bigwedge_{k=1, \dots, n} (\mathbf{G} \mathbf{F} \text{enabled}_k \Rightarrow \mathbf{G} \mathbf{F} \text{executed}_k)$$



Faire Kripke Strukturen

Hier zwei Beispiele:

- „eine Alternative einer sich ständig wiederholenden Alternative wird irgendwann einmal auch gewählt“ z.B. Hardware Arbitrer
- „ein gestörter Kanal übermittelt immer wieder einmal eine Nachricht korrekt“ z.B Alternierbitprotokoll

LTL

ausdrückbar in CTL^* , aber *nicht* in CTL . CTL ist erwünscht wegen besserer Komplexitäts-Eigenschaften bei der Analyse.

Ausweg: „faire Semantik“

Fairness-Eigenschaften oft durch Zustandsmengen ausdrückbar:

Faire Kripke Strukturen

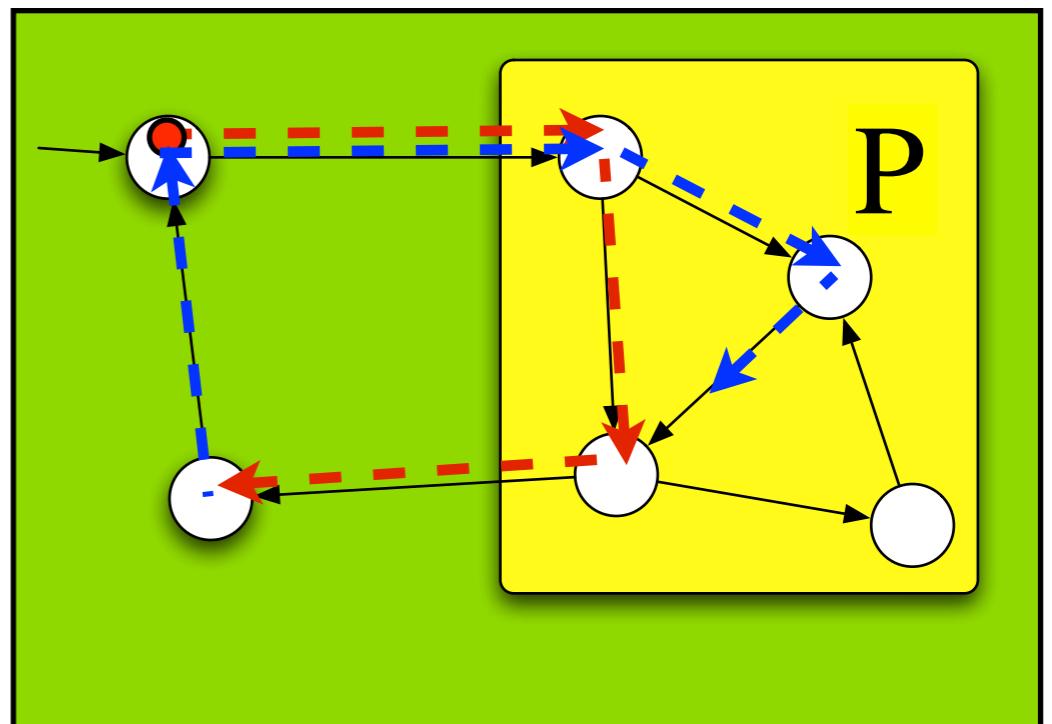
Definition: Eine *faire* Kripke-Struktur

$$M := (S, S_0, R, E_S, \{E_F^1, \dots, E_F^k\})$$

besteht aus einer Kripke-Struktur $M := (S, S_0, R, E_S)$ und den $k \geq 1$ Endzustandsmengen $E_F^i \subseteq S$.

Ein Pfad $\pi = s_0s_1s_2 \dots \in SS(M)$ heißt *fair*, falls $\text{infinite}(\pi) \cap E_F^i \neq \emptyset$ für alle $i \in \{1, \dots, k\}$ gilt

Fairness wird also über eine Akzeptanzbedingung definiert, die wir bereits vom Büchi-Automaten kennen.



FGI 2

Daniel Moldt

Kap. 7: Systemverifikation durch Petrinetze

FGI 2

Daniel Moldt

Zustandsraumexplosion

Größe des Zustandsraumes

Es gibt Petri-Netze mit im Verhältnis zu ihrer Größe sehr großer Erreichbarkeitsmenge, wie das folgende Ergebnis zeigt. Diese Erscheinung ist unter dem Begriff *Zustandsexplosion* bekannt.

Satz 3.27 *Es gibt eine unendliche Folge von beschränkten Petri-Netzen $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \dots$, deren Größe ($|P| + |T| + |F|$) linear wächst, jedoch die Größe der Erreichbarkeitsmengen $|\mathbf{R}(\mathcal{N}_1)|, |\mathbf{R}(\mathcal{N}_2)|, |\mathbf{R}(\mathcal{N}_3)|, \dots$ wächst schneller als jede primitiv rekursive Funktion.*

$$2^{2^n} \quad 2^{2^{2^n}} \quad \dots \quad 2^{2^{2^{2^{2^{2^{2^n}}}}}}$$

Hyper-exponentielles Wachstum

```
function hexp(n) is
    val := 1
    for i := 1 to n do
        cnt := val
        while cnt > 0 do
            val := 2 * val
            cnt := cnt - 1
        endwhile
    endfor
    return val
```

$2^{2^{2^{2^{2^{2^{2^{2^n}}}}}}}$

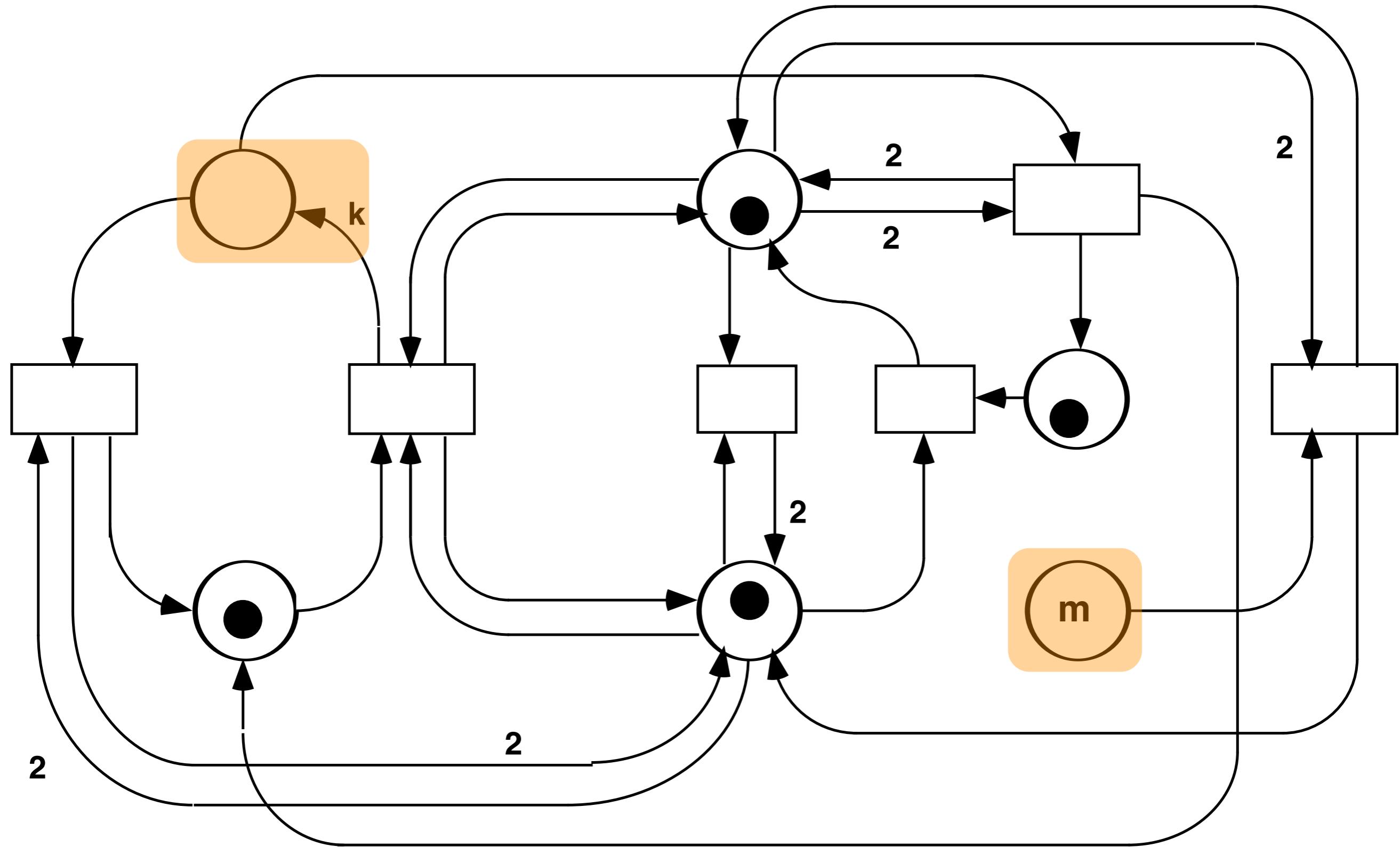
Allgemein gilt die Rekursionsgleichung:

$$val_{i+1} = val_i \cdot 2^{val_i} \quad \text{und} \quad val_0 = 1$$

Wir wollen das Wachstum von val_i studieren und schätzen daher $val_i \cdot 2^{val_i}$ durch $v_i := 2^{v_i}, v_0 = 1$ von unten ab. Es ist $v_0 = 1, v_1 = 2, v_2 = 2^2, v_3 = 2^{2^2}, v_4 = 2^{2^{2^2}}$ usw. Abwickeln der Rekursion ergibt also:

$$v_n = 2^{v_{n-1}} = 2^{2^{v_{n-2}}} = 2^{2^{2^{v_{n-3}}}} = 2^{2^{\dots^2}} \} \text{n-mal}$$

“Berechnung” von val



“Berechnung” von val

Das P/T-Netz \mathcal{N}_7 hat 6 Plätzen, 6 Transitionen, $30 + k$ Kanten und $m + 4$ Marken in der eingezeichneten Anfangsmarkierung \mathbf{m}_0 . Die Größen $k \geq 2$ und $m \geq 0$ sind festlegbar und die maximale Zahl von möglichen Marken bestimmt sich durch die Funktion

$$\max(m, k) := k \cdot f_k(m) + 2,$$

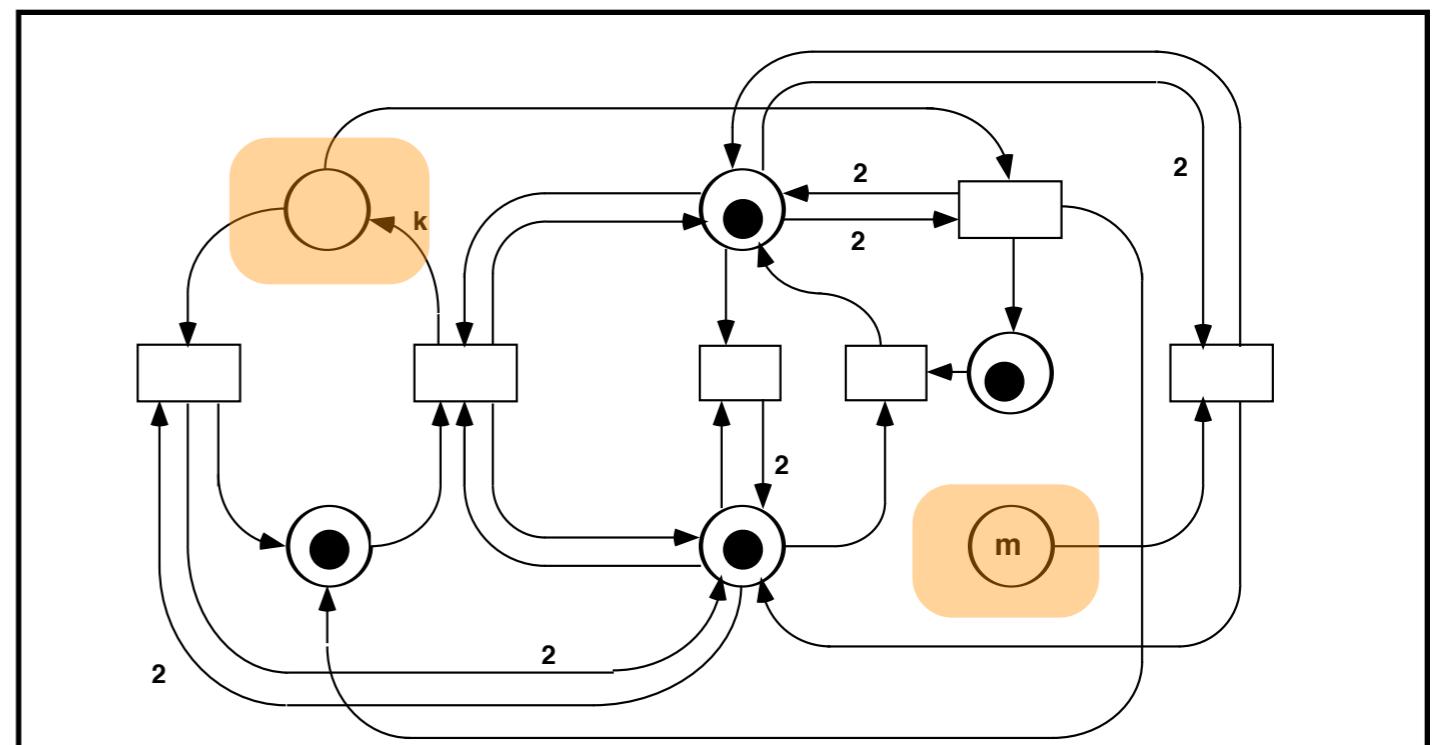
wobei f_k wie folgt definiert ist:

$$f_k(m) := \text{IF } m = 0 \text{ THEN } k \text{ ELSE } f_k(m - 1) \cdot k^{f_k(m-1)} \text{ FI.}$$

Es gilt: $\max(2, 2) = 4098$

$$\max(3, 2) = 1048576^{103} + 2$$

$$\max(2, 3) = 3^{86} + 2.$$



Eigenschaften von Überdeckungsgraphen (Wdh.)

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.

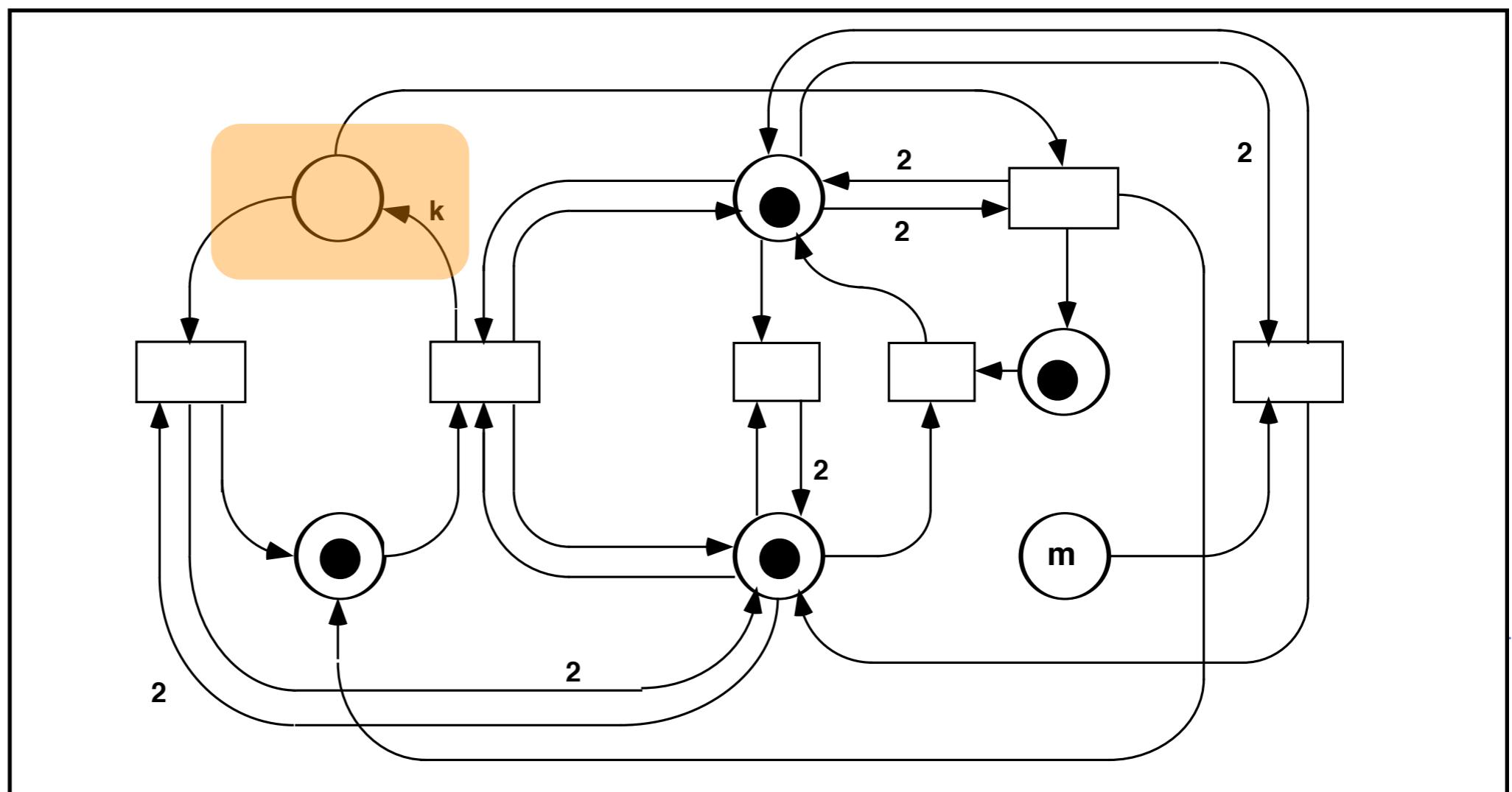
Satz 7.17 Der Algorithmus 7.4 zur Konstruktion von $G(\mathcal{N})$ terminiert.

Theorem: 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.

Zustandsraumexplosion

Dieses Netz zeigt, dass eine alleinige Beschränkung der Kapazität der Plätze in der Regel nicht ausreicht, um niedrige Komplexitätseigenschaften zu erreichen, denn das Netz \mathcal{N}_7 hat stets eine endliche Erreichbarkeitsmenge, diese ist jedoch so groß, dass alle Petrinetz-Analysetools auf ihre Grenzen hin getestet werden können.

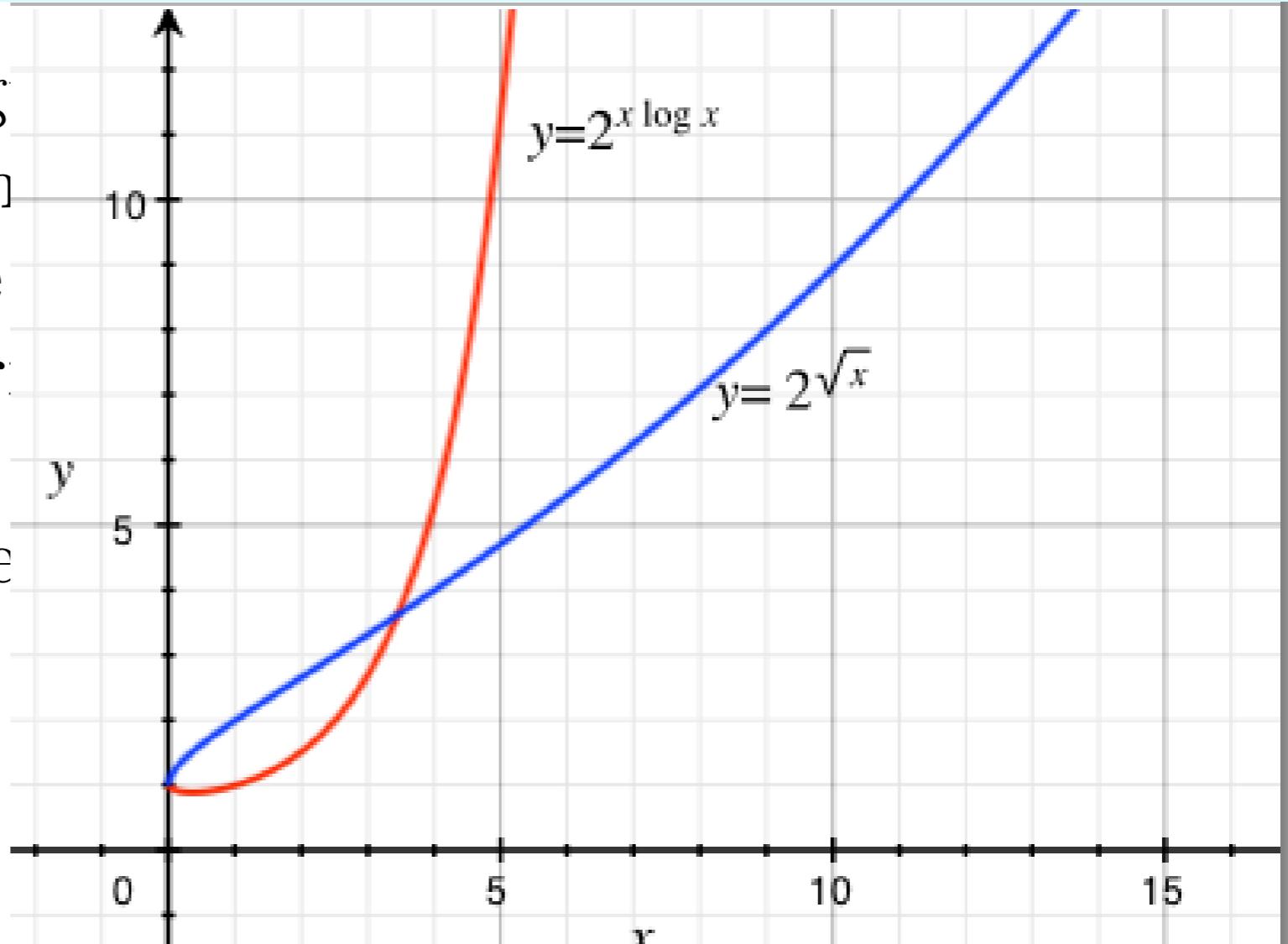


$$\max(2, 3) = 3^{86} + 2.$$

$$\max(3, 2) = 1048576^{103} + 2$$

Komplexität

Aus diesem Ergebnis folg
scheidungsverfahren (Kon
der Erreichbarkeitsmenge
deckungsgraphen nicht pr
die Frage, ob es vielleicht
dieses Problem entschiede



Satz 3.28 (Rackoff (1978)) Das Beschränktheitsproblem ist mit $O(2^{c \cdot n \cdot \log(n)})$ Platzbedarf entscheidbar.

Satz 3.29 (Lipton (1976)) Das Beschränktheitsproblem benötigt für seine Entscheidung mindestens $O(2^{c \cdot \sqrt{n}})$ Platzbedarf.

Komplexität

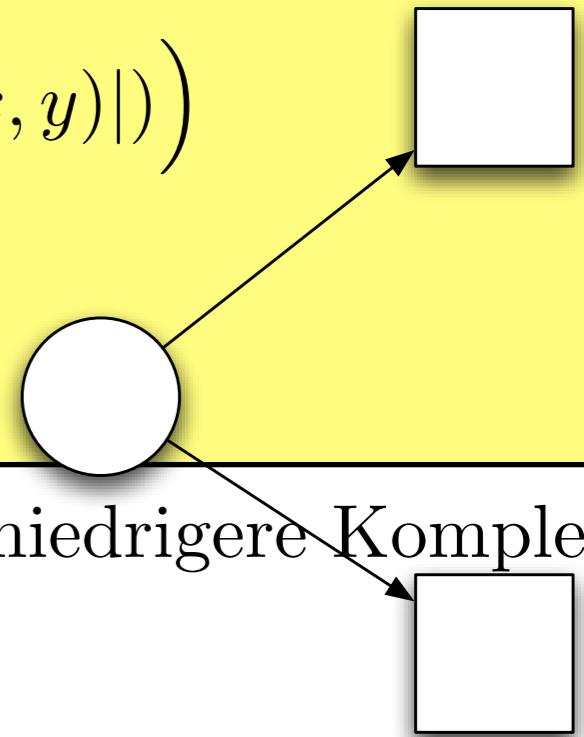
Ein besseres Ergebnis ist von Rosier und Yen bewiesen worden:

Satz: [Rosier und Yen, 1986] Das Beschränktheitsproblem kann für ein gegebenes P/T-Netz $\mathcal{N} = \langle P, T, F, W, \mathbf{m}_0 \rangle$ mit

$$O\left(2^{c \cdot |P| \cdot \log(|P|)} \cdot (\log(|T|) + \max_{x,y \in P \cup T} (|W(x,y)|))\right)$$

Platzbedarf entschieden werden.

Für festes $|P|$ ist das Problem PSPACE-vollständig.



Erst für spezielle Teilklassen der Petrinetze kann man eine niedrigere Komplexität für dieses Entscheidungsproblem erwarten.

Definition: Ein P/T-Netz $\mathcal{N} = \langle P, T, F, W, \mathbf{m}_0 \rangle$ heißt *konfliktfrei*, falls es keinen *Konfliktplatz* enthält, d.h. einen Platz p mit $|p^\bullet| > 1$.

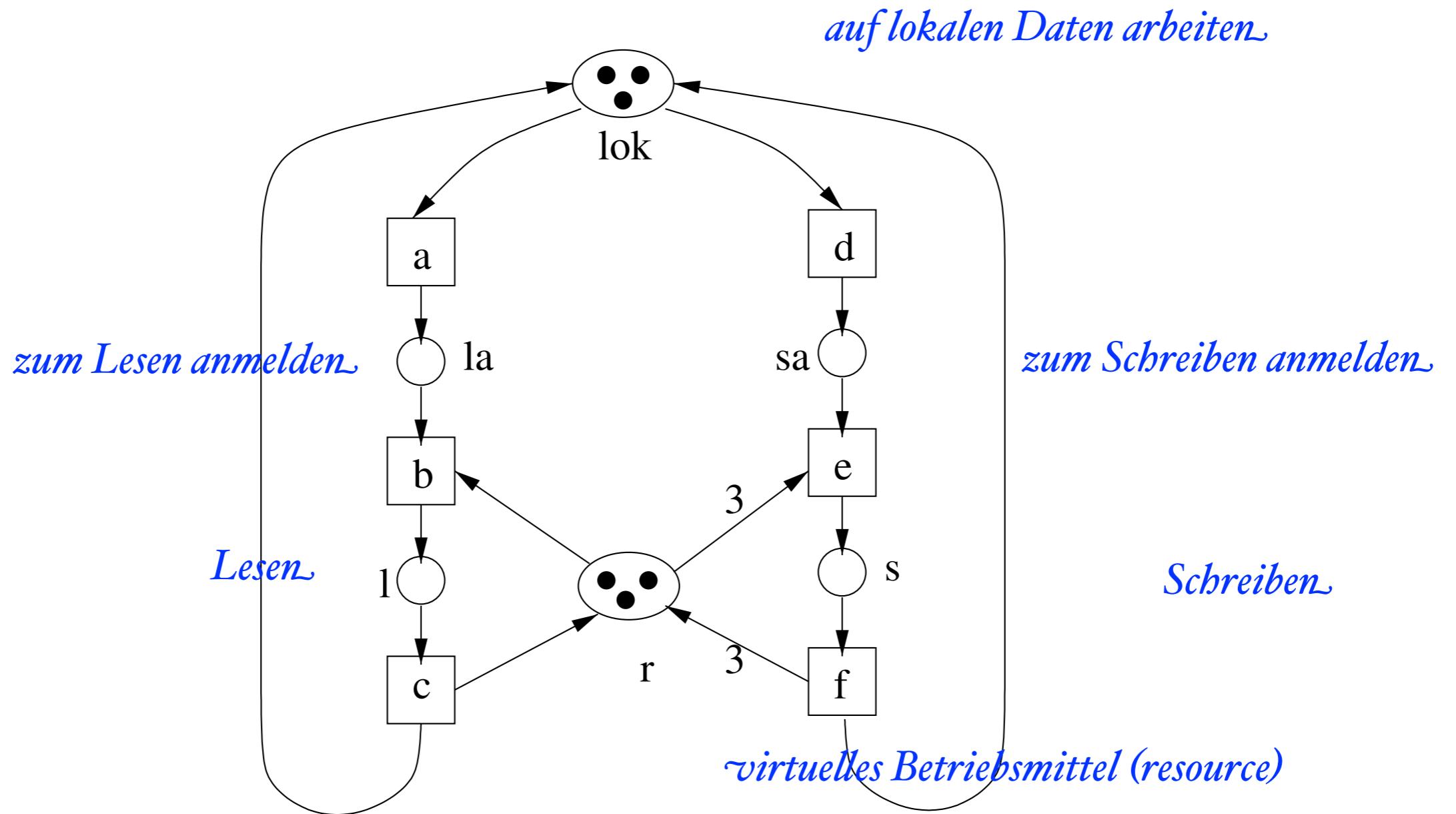
Satz: [Rosier et. al., 1987] Das Beschränktheitsproblem kann für konfliktfreie Petrinetze mit $O(n^{1,5})$ Platzbedarf entschieden werden.

FGI 2

Daniel Moldt

**Strukturelle Eigenschaften:
Netzinvarianten, Fallen**

“Leser-Schreiber-Problem”

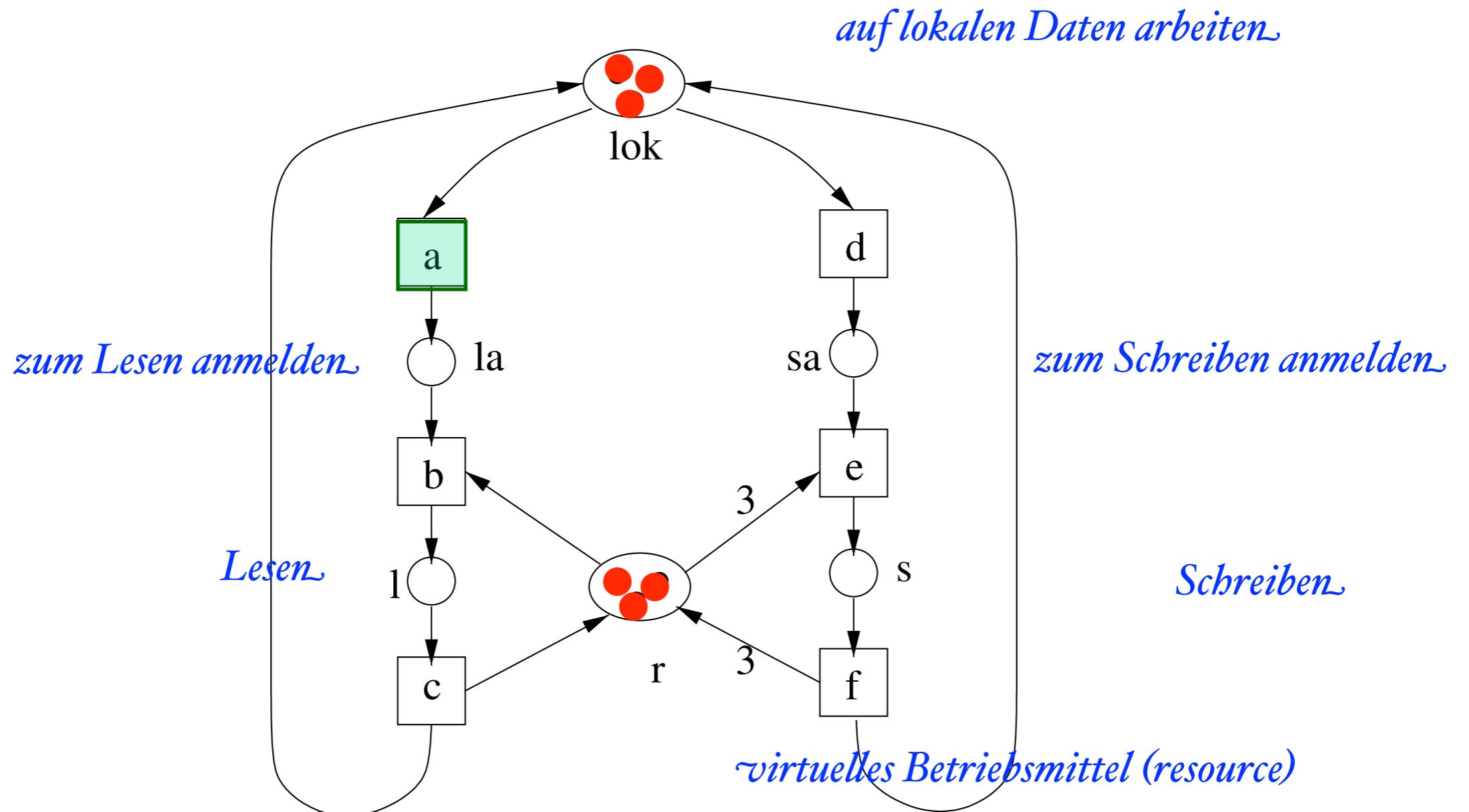


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) a a d e f b b c c a

2.) a d a e f b c a b c

3.) d a a e f b b c a c

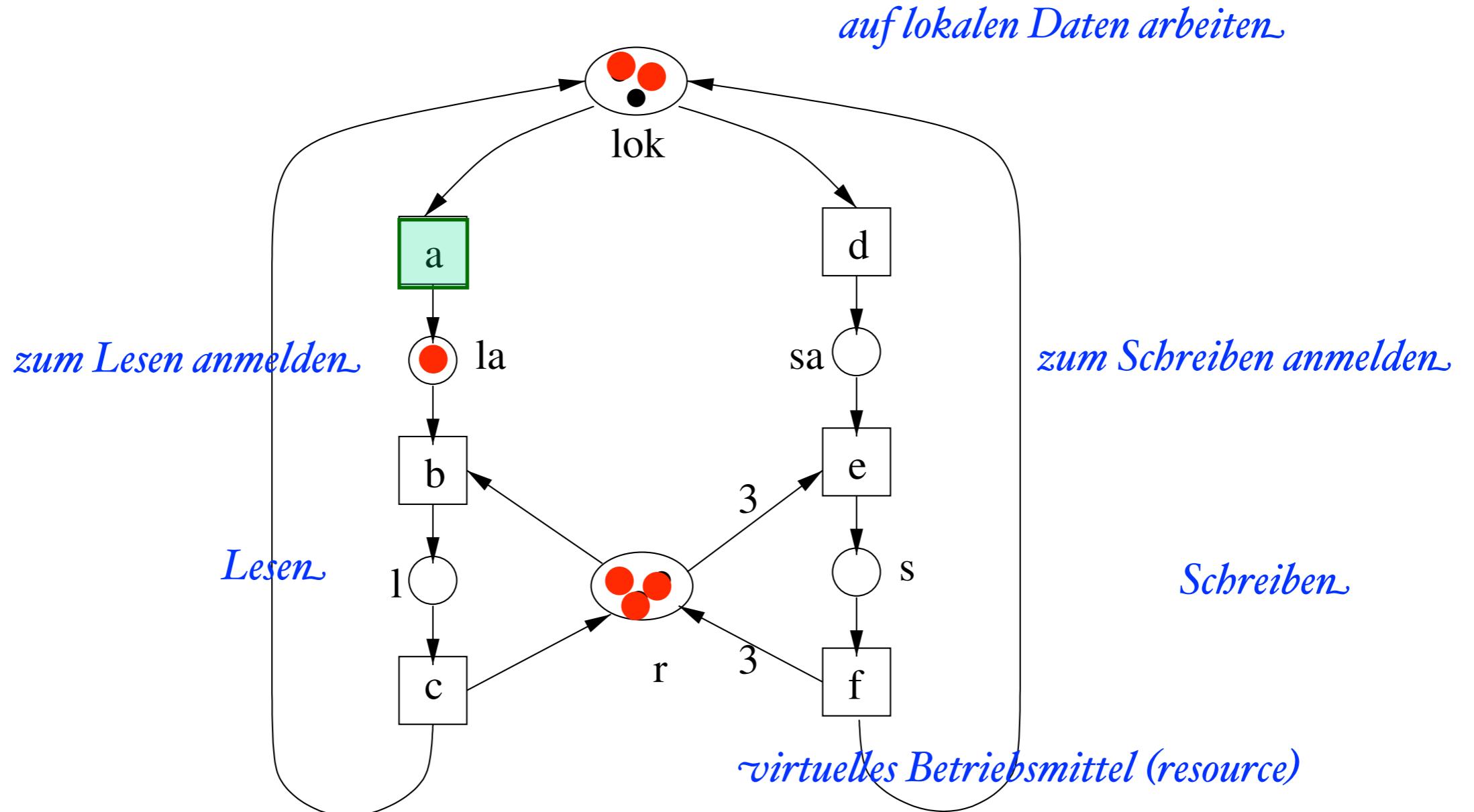


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) a a d e f b b c c a

2.) a d a e f b c a b c

3.) d a a e f b b c a c

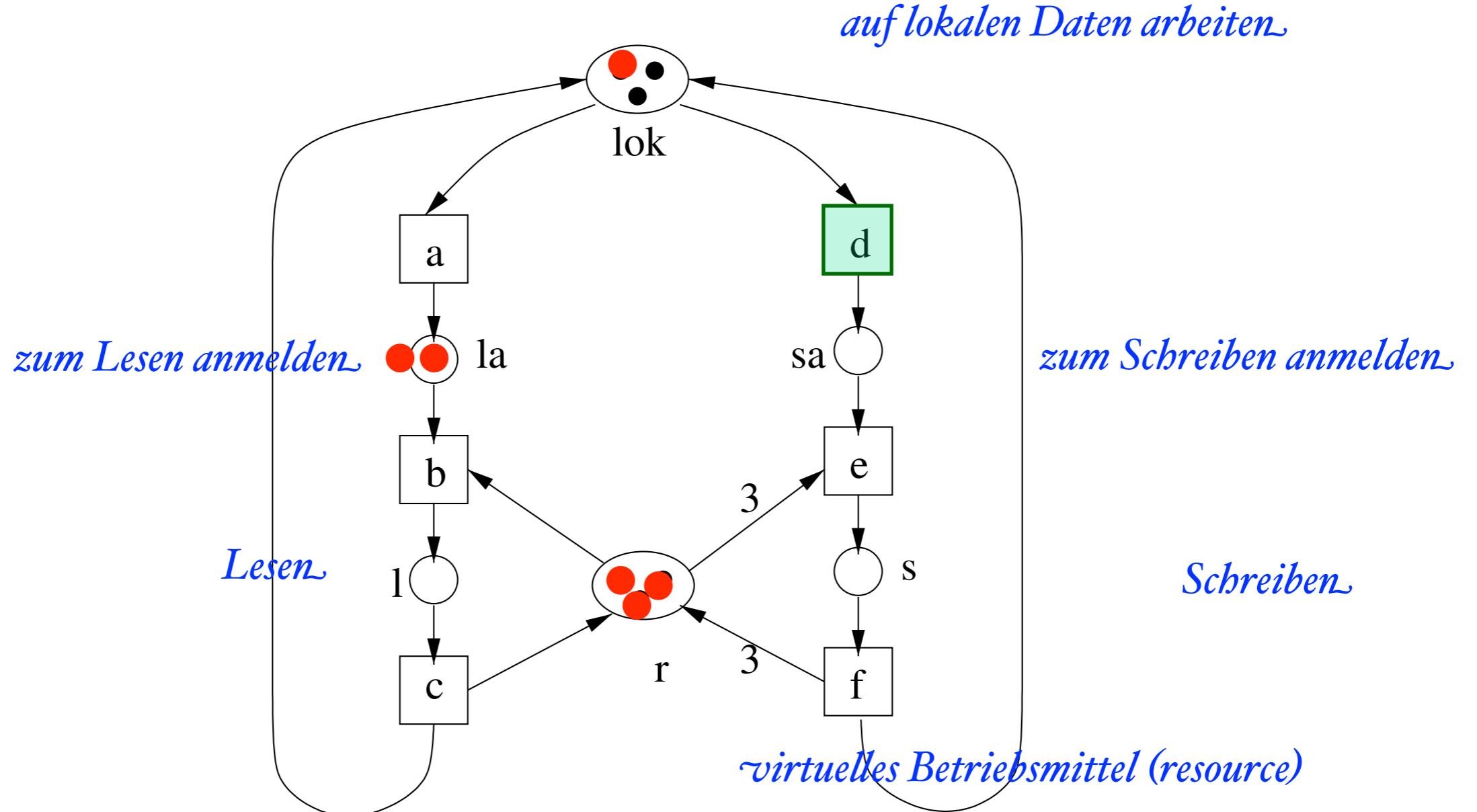


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \underline{[a]} d e f b b c c a$

2.) $a d a e f b c a b c$

3.) $d a a e f b b c a c$

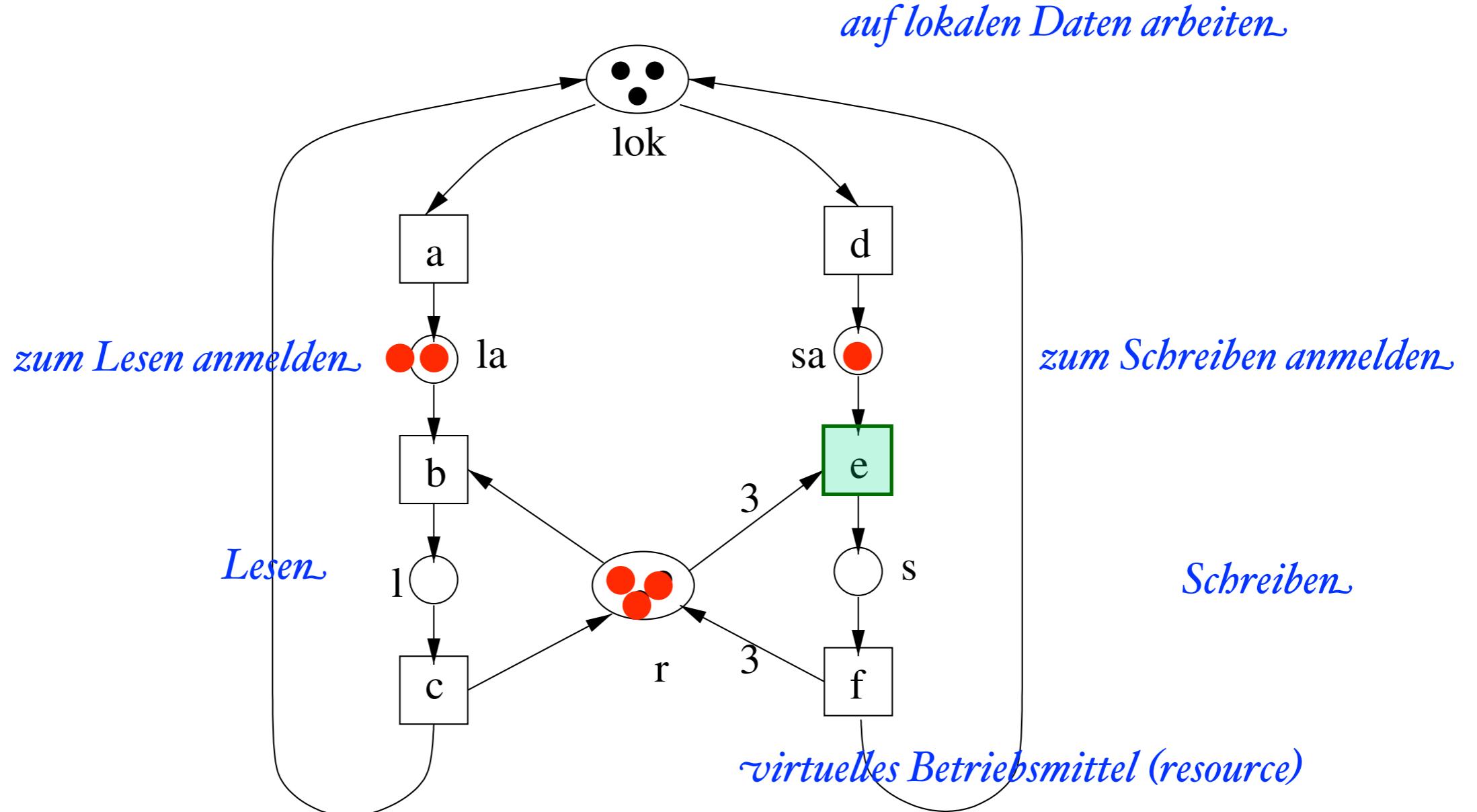


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \underline{a} \underline{d} e f b b c c a$

2.) $a d a e f b c a b c$

3.) $d a a e f b b c a c$

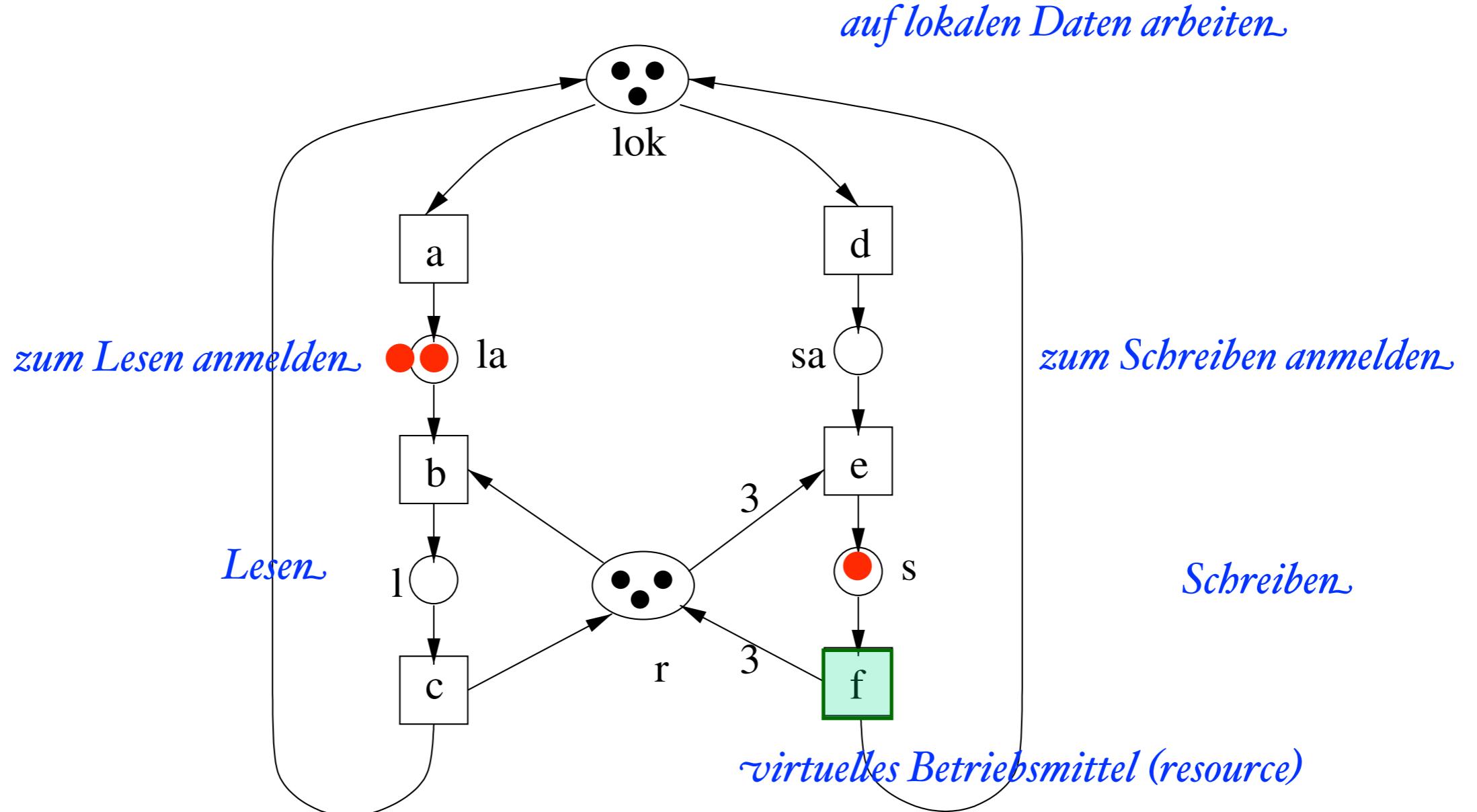


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d [e] f b b c c a$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

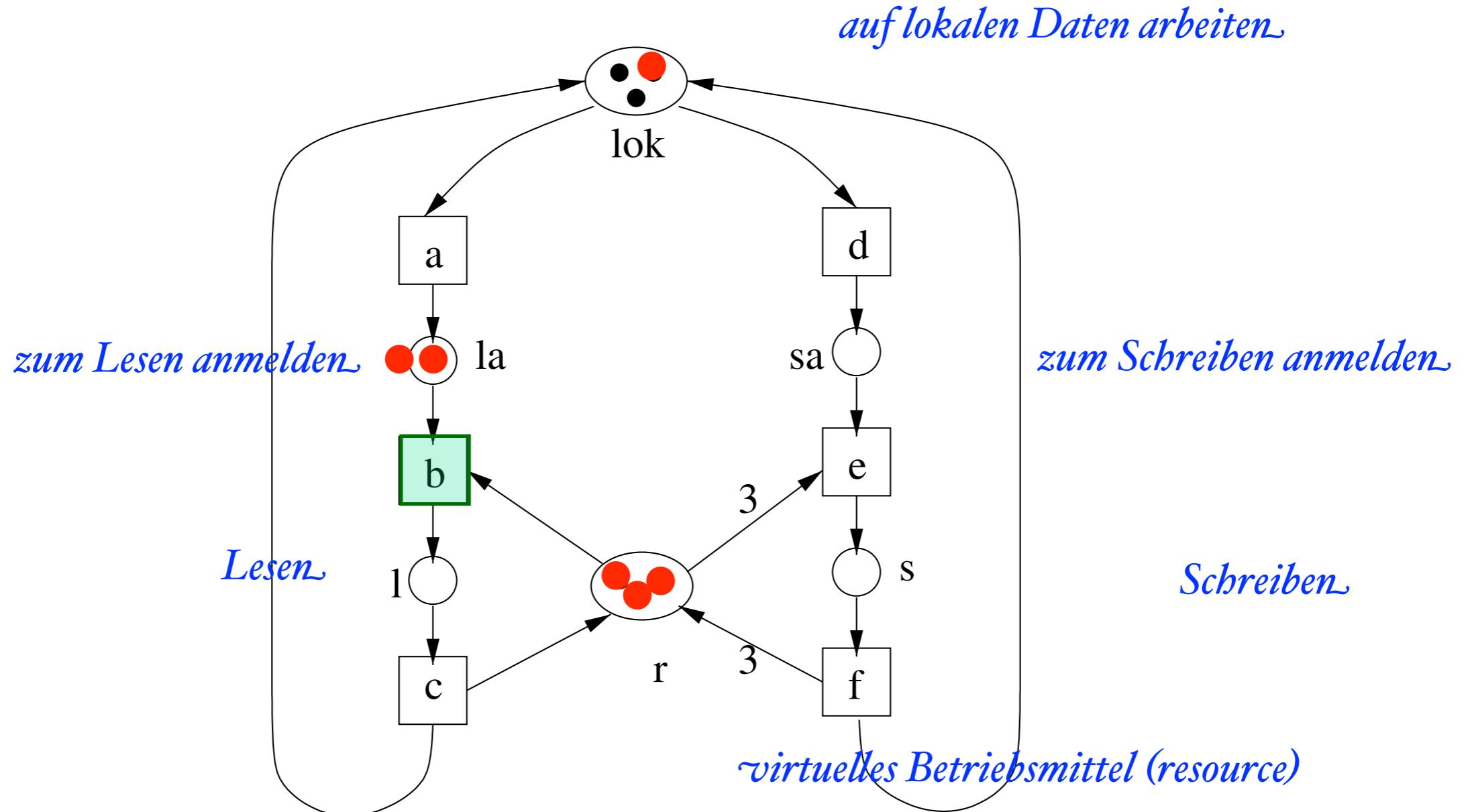


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \ b \ b \ c \ c \ a$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

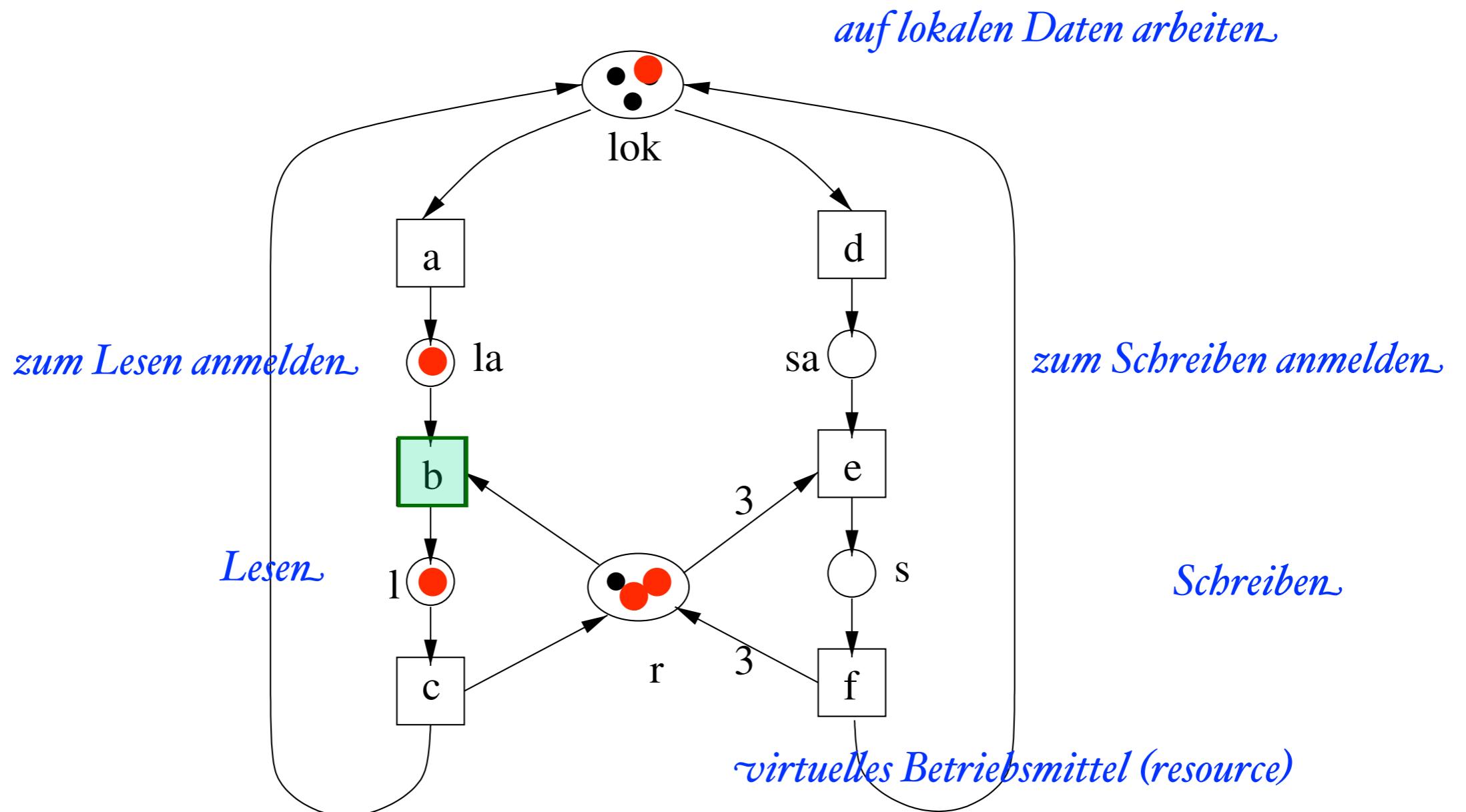


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \underline{b} \underline{b} \underline{c} \underline{c} \underline{a}$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

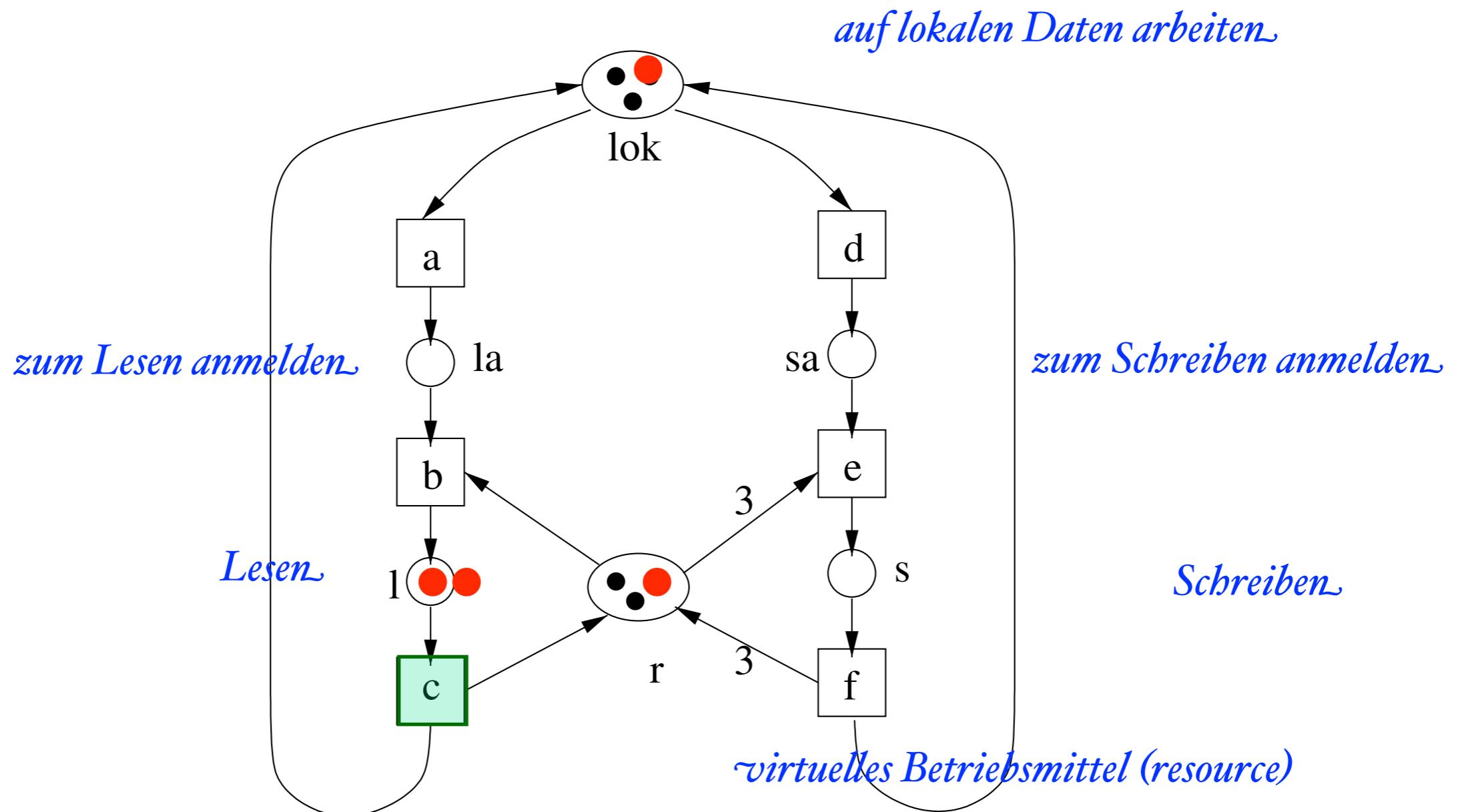


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \ b \underline{b} \underline{c} \underline{c} \ a$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

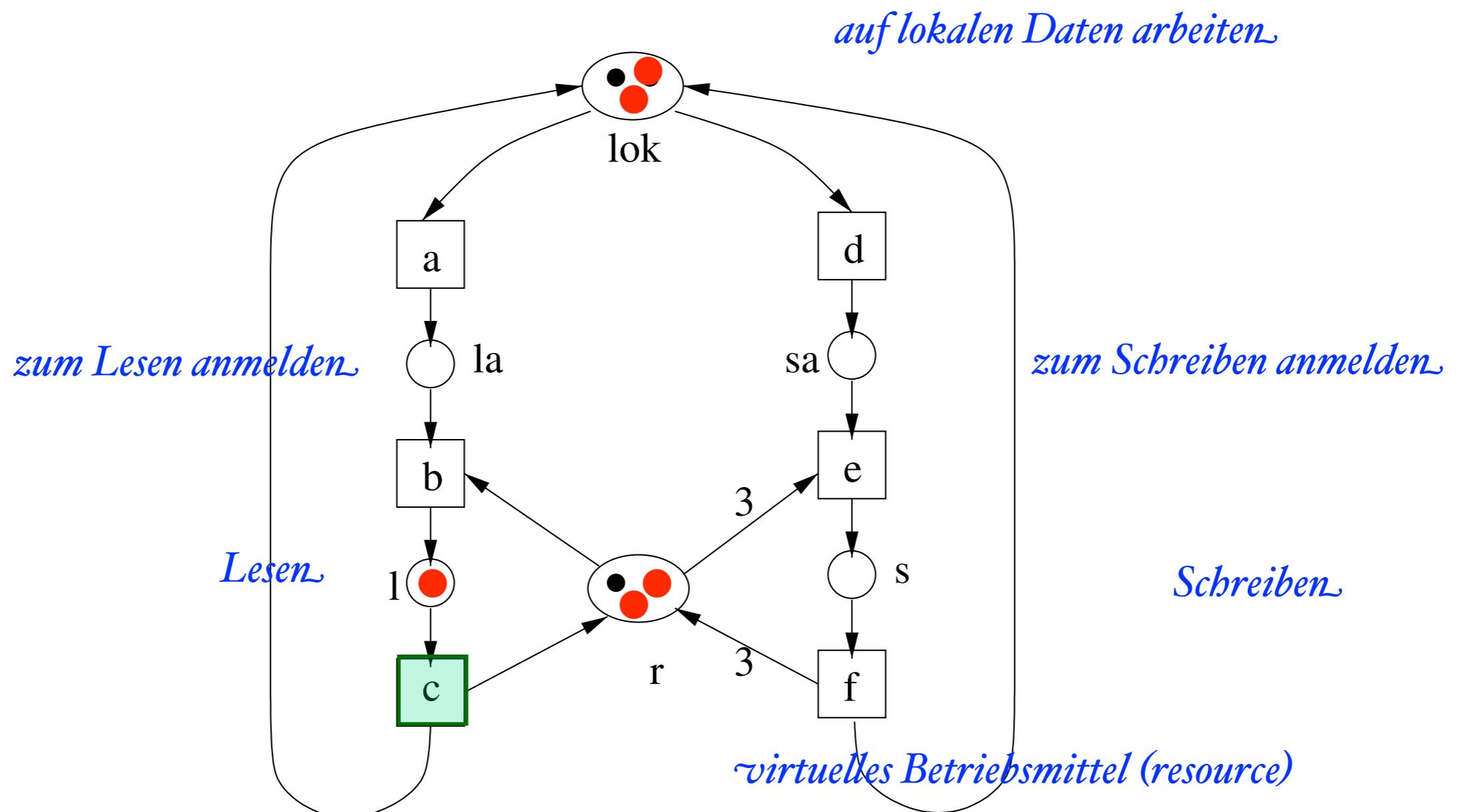


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \ b \ b \underline{c} \underline{c} \ a$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

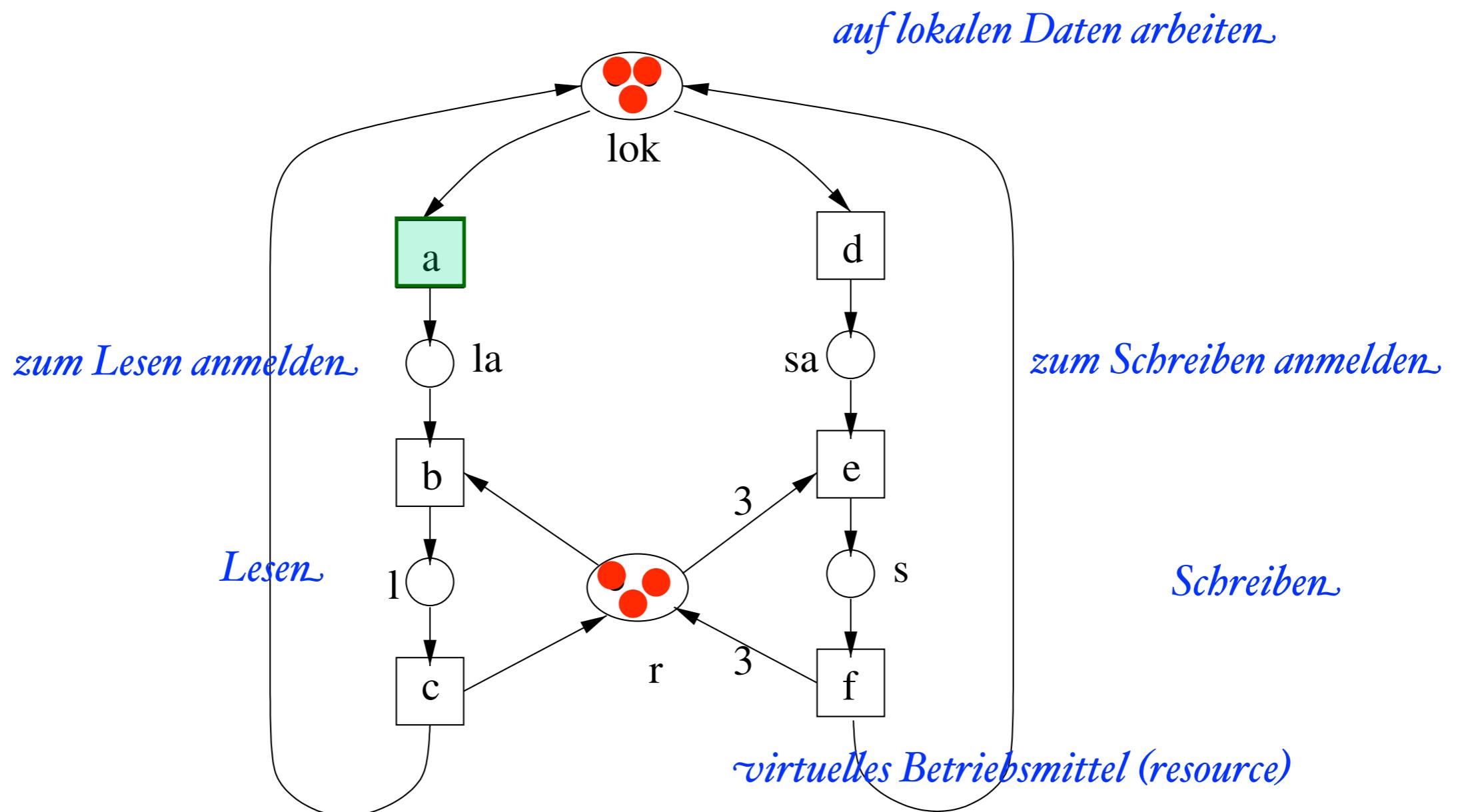


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \ b \ b \ c \underline{c} \underline{a}$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

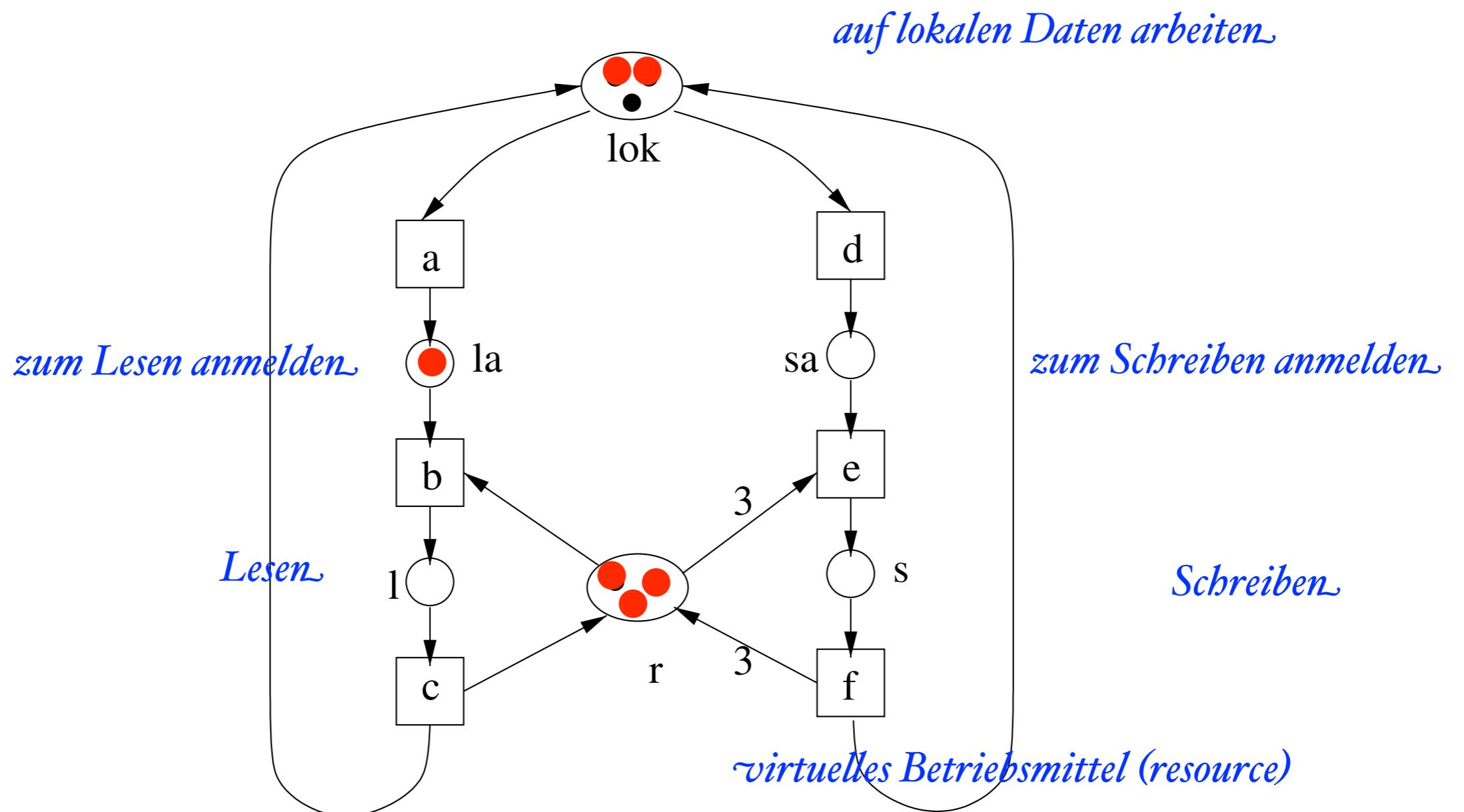


Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \ b \ b \ c \ c \underline{a}$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$



Die Auftragsbeschreibungen können z. B. so beschaffen sein, dass folgende serielle Prozesse ablaufen :

1.) $a \ a \ d \ e \ f \ b \ b \ c \ c \ a$

2.) $a \ d \ a \ e \ f \ b \ c \ a \ b \ c$

3.) $d \ a \ a \ e \ f \ b \ b \ c \ a \ c$

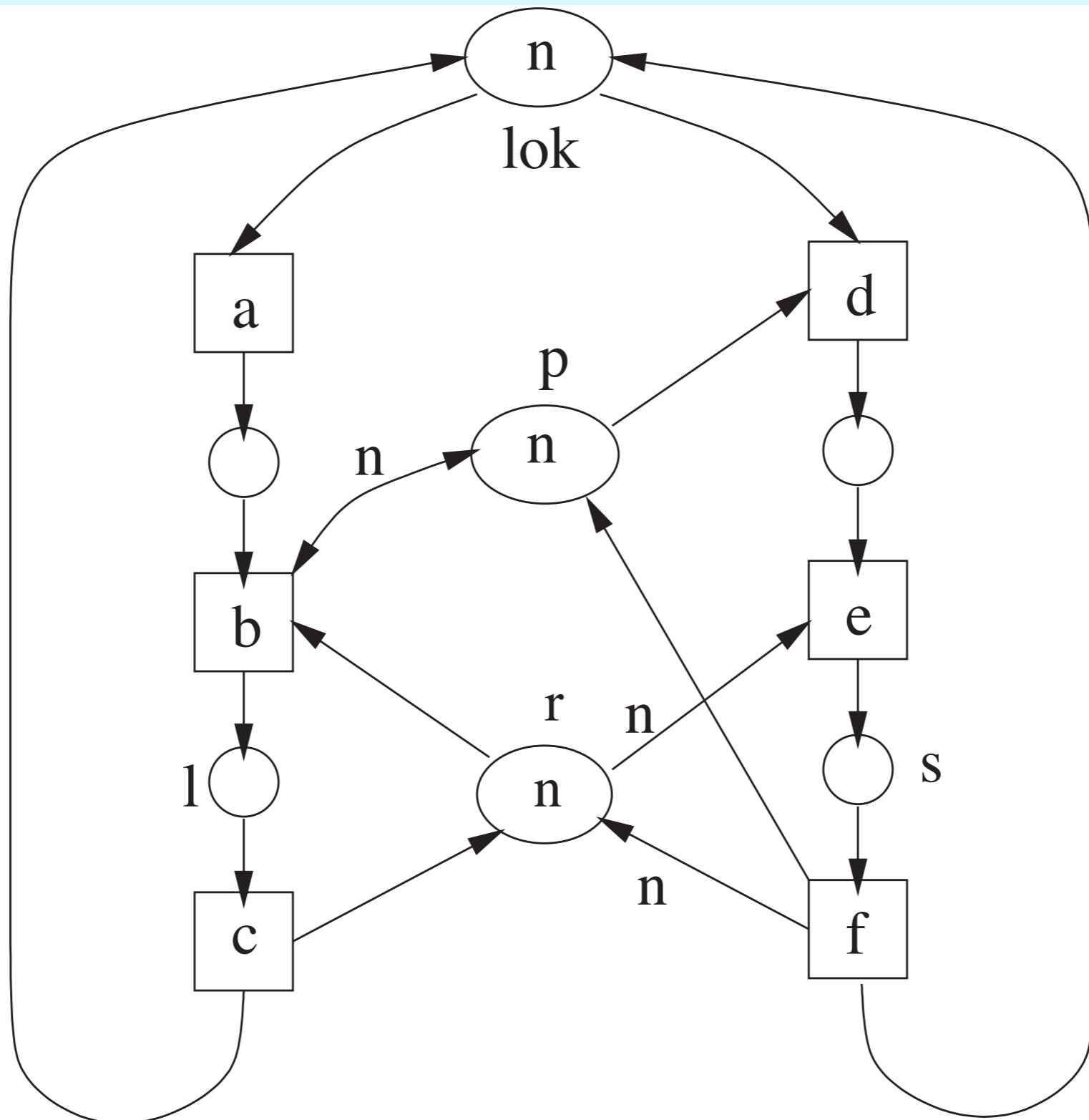
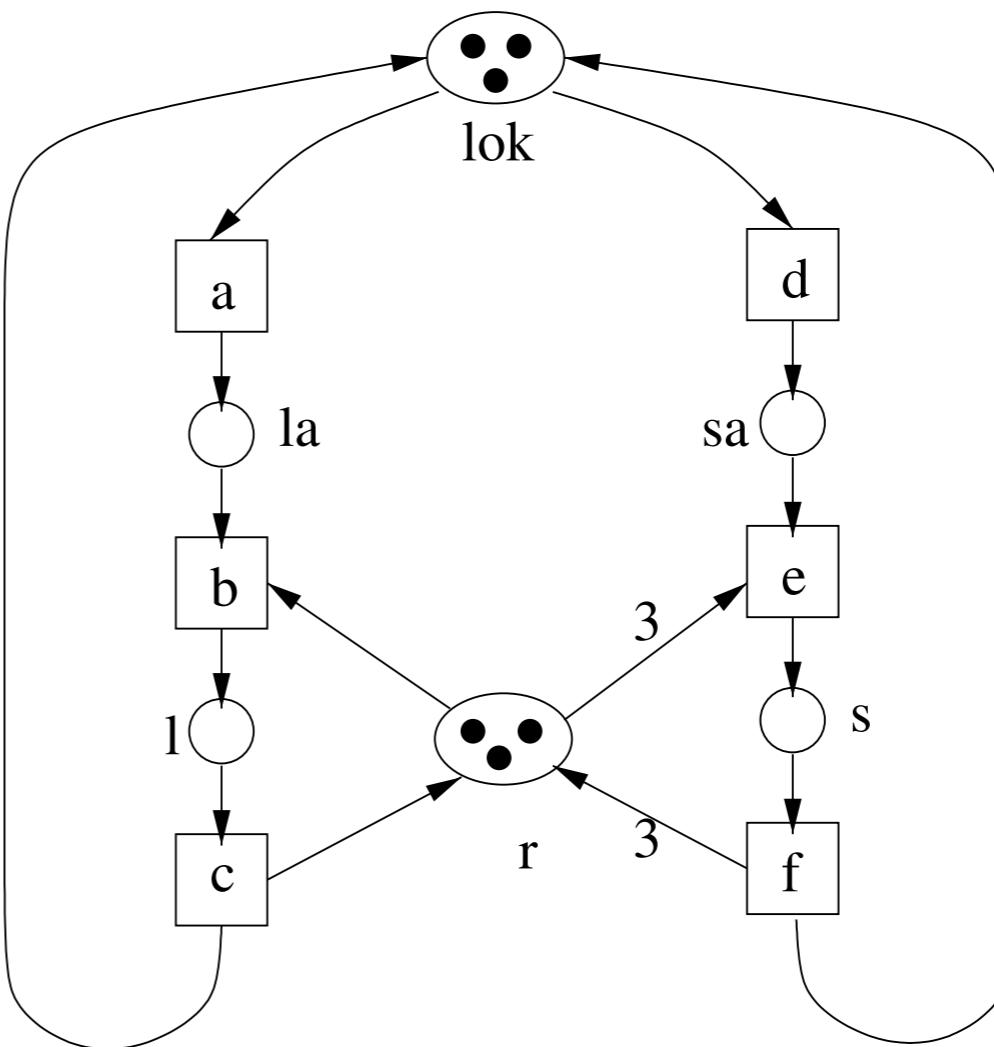


Abbildung 5.15: Priorität für Schreibaufträge

Spezifikation

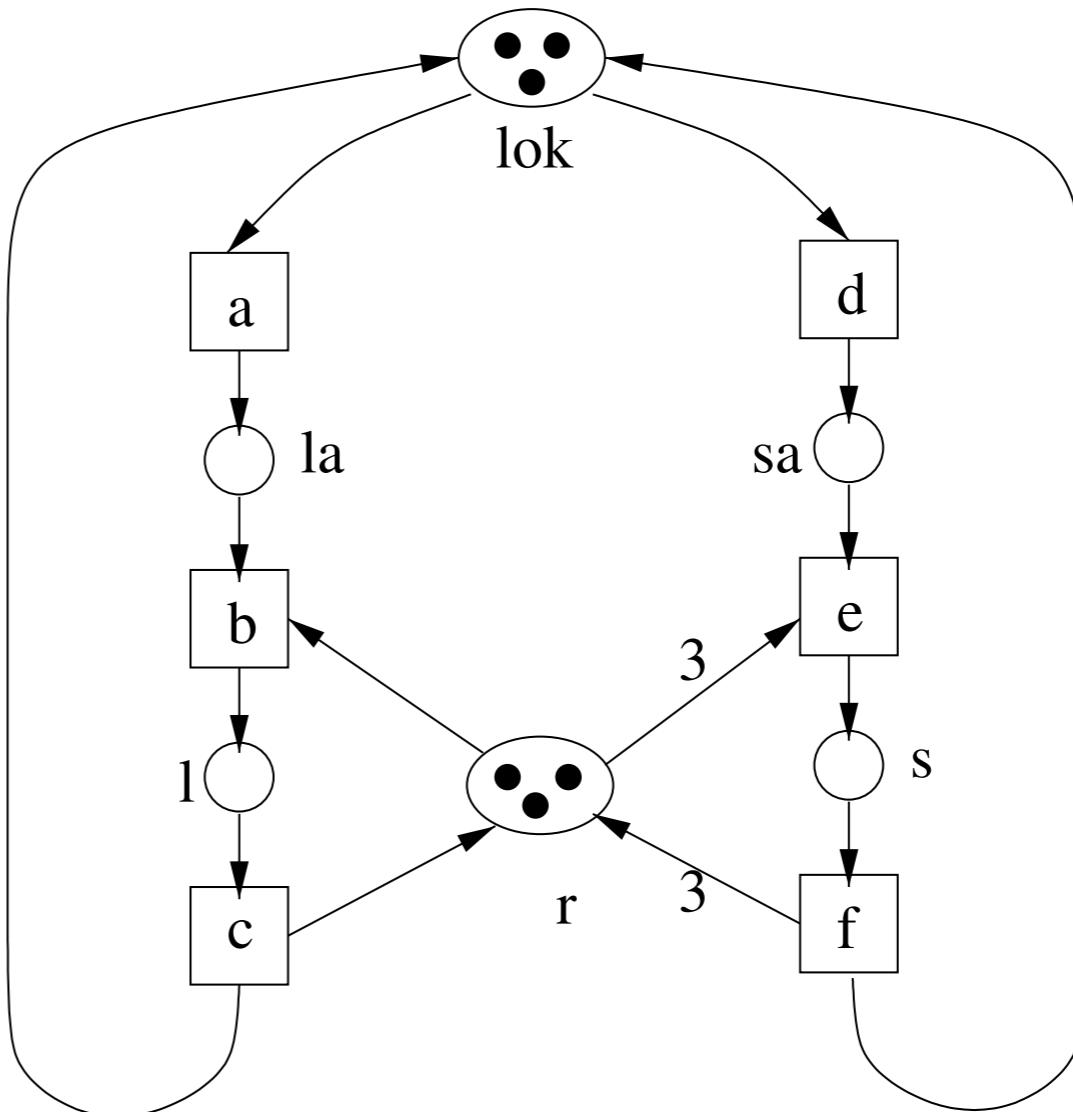


Für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ soll also gelten:

- a) $\mathbf{m}(s) > 0 \rightarrow \mathbf{m}(l) = 0$
- b) $\mathbf{m}(s) \leq 1$
- c) $\mathbf{m}(l) > 0 \rightarrow \mathbf{m}(s) = 0$

- a) Wenn ein Schreiber schreibt, darf kein Leser lesen
- b) Es darf höchstens ein Schreiber schreiben
- c) Wenn ein Leser liest, darf kein Schreiber schreiben

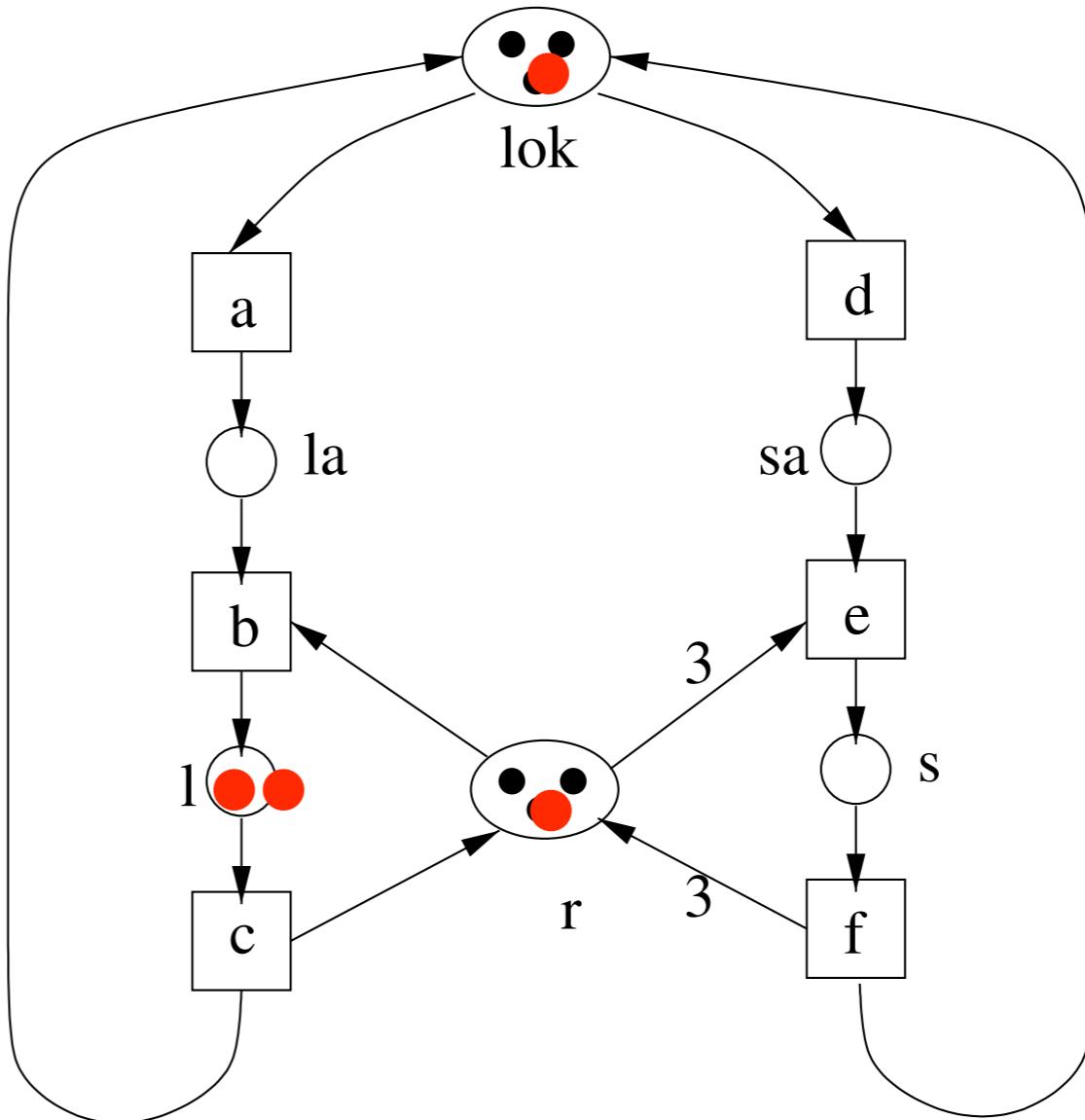
Spezifikation



- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + \underset{3}{n} \cdot s = \underset{3}{n}$

(lok, la, \dots) stehen hier abkürzend für $\mathbf{m}(lok), \mathbf{m}(la), \dots$

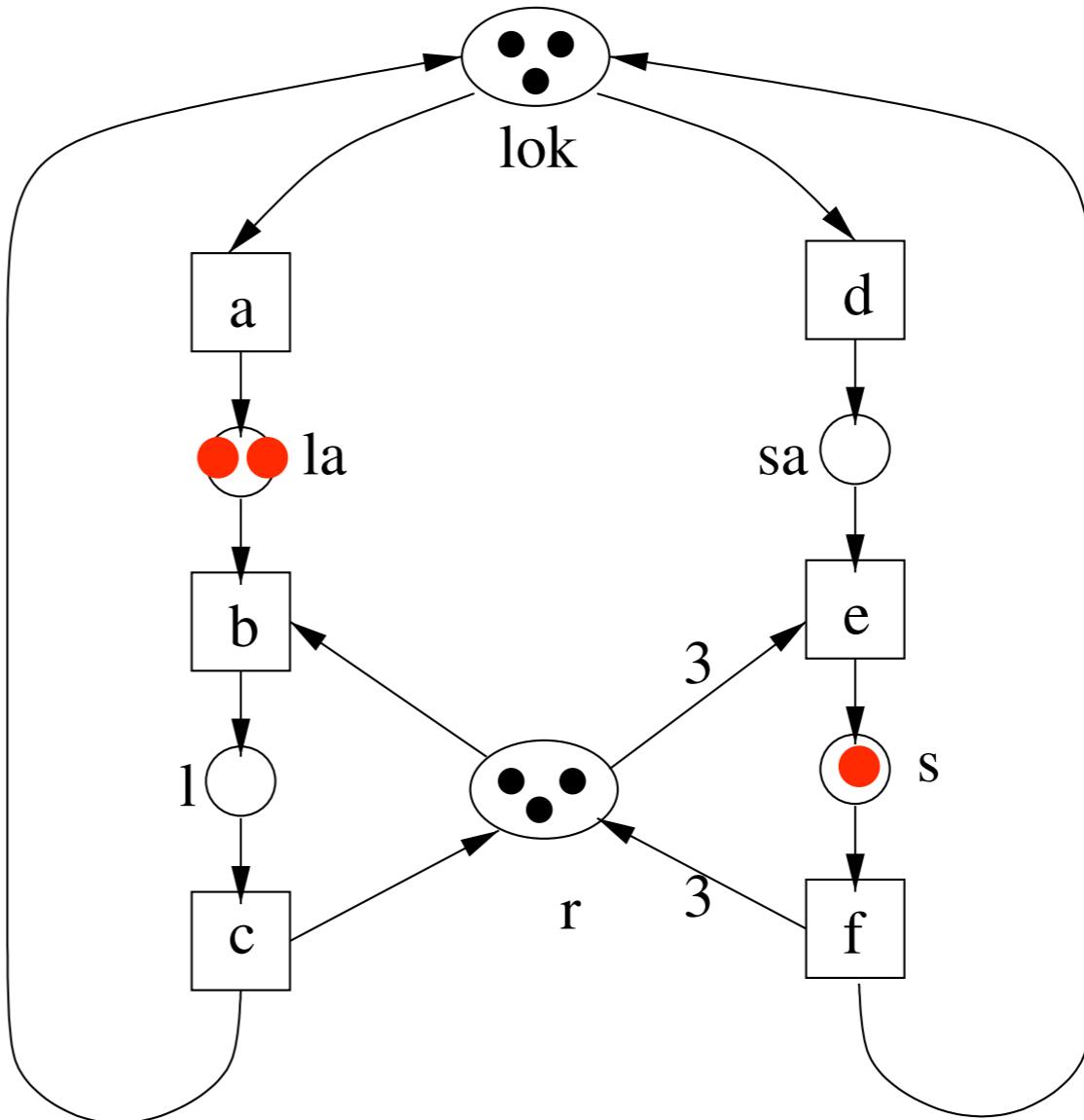
Spezifikation



- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + \frac{n}{3} \cdot s = n$

(lok, la, \dots) stehen hier abkürzend für $\mathbf{m}(lok), \mathbf{m}(la), \dots$

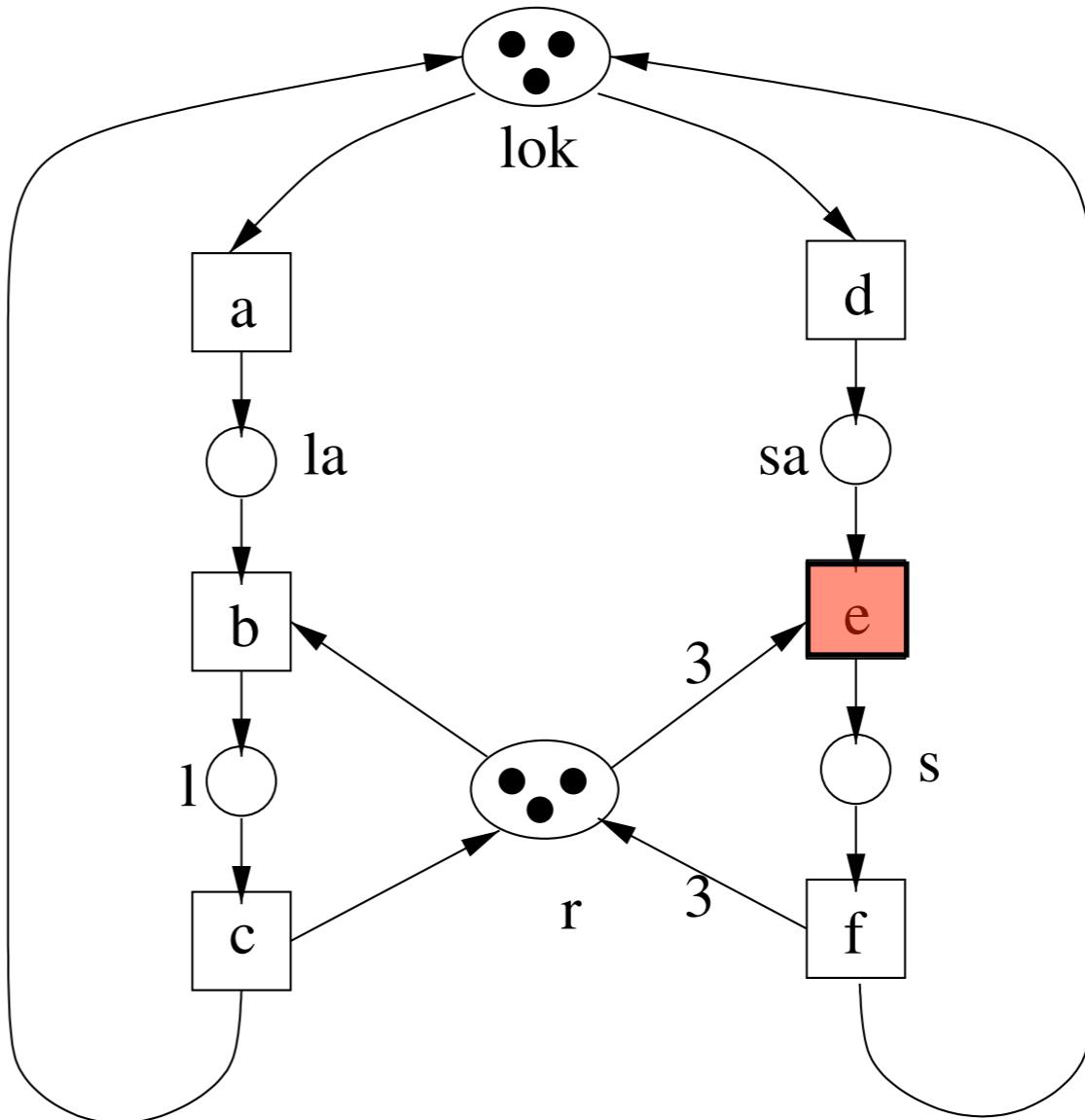
Spezifikation



- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + \frac{n}{3} \cdot s = n$

(lok, la, \dots) stehen hier abkürzend für $\mathbf{m}(lok), \mathbf{m}(la), \dots$

Spezifikation

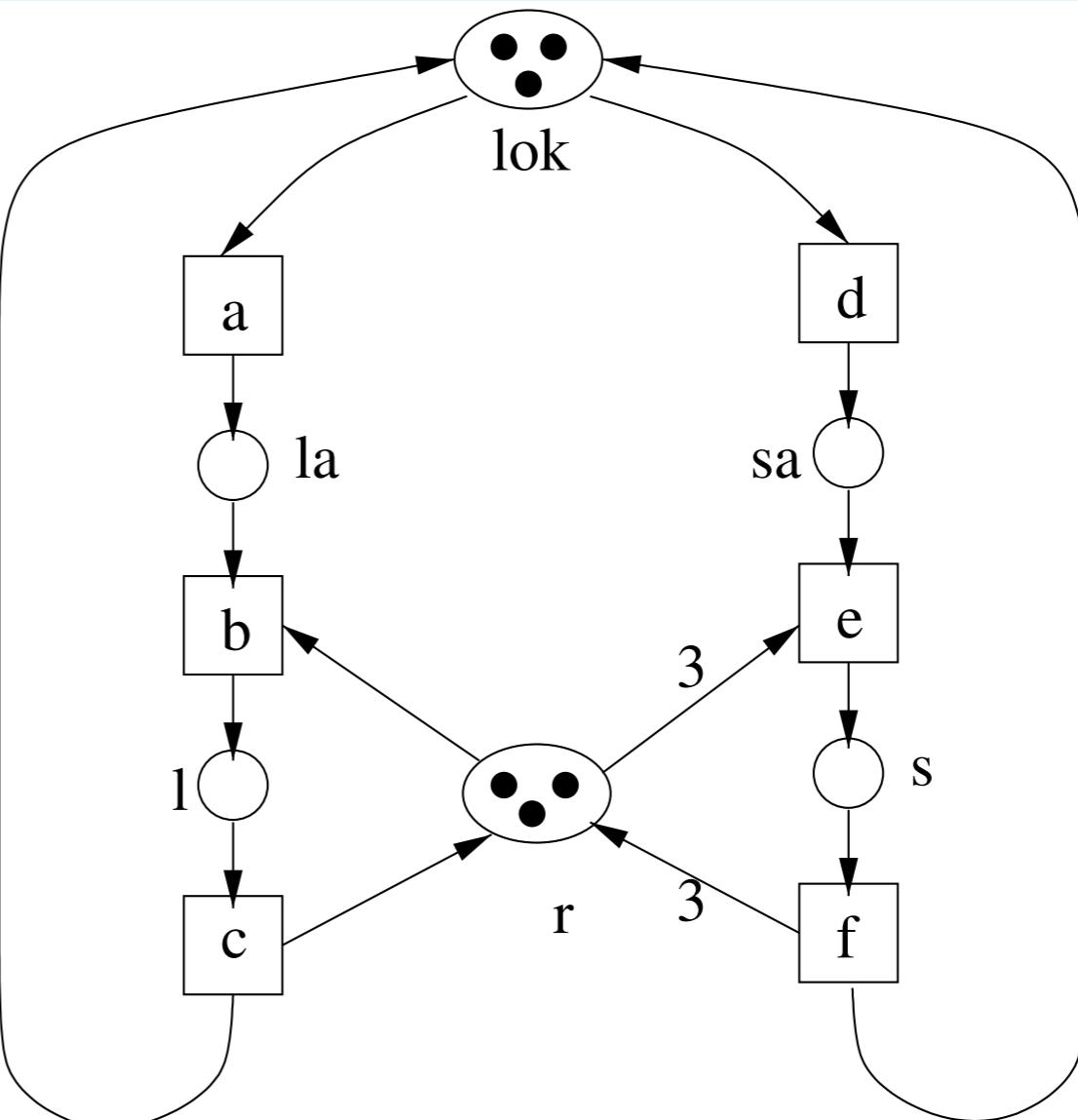


- $i_1 : lok + la + sa + l + r + s = n$
- $i_2 : l + r + n \cdot s = n$

$$\begin{array}{ccc} -1 & & +1 \\ \textcolor{red}{-3} & & \textcolor{red}{+1} \end{array}$$

(lok, la, \dots) stehen hier abkürzend für $\mathbf{m}(lok), \mathbf{m}(la), \dots$

Spezifikation



Für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ soll also gelten:

- a) $\mathbf{m}(s) > 0 \rightarrow \mathbf{m}(l) = 0$
- b) $\mathbf{m}(s) \leq 1$
- c) $\mathbf{m}(l) > 0 \rightarrow \mathbf{m}(s) = 0$



- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + n \cdot s = n$

S-Invarianten-Gleichungen

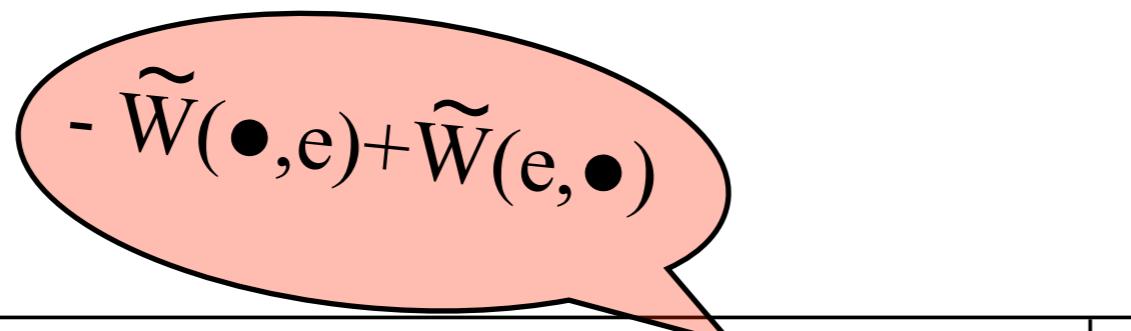
- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + n \cdot s = n$

Definition: Es sei $\mathcal{N} = \langle S, T, F, W, \mathbf{m}_0 \rangle$ ein S/T -Netz mit $S = \{s_1, \dots, s_p\}$. Eine Gleichung der Form $\sum_{i=1}^p k_i \cdot \mathbf{m}(s_i) = k$ mit $k_i, k \in \mathbb{Z}$ heißt *S-Invarianten-Gleichung*, wenn sie für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ gilt. Abkürzend wird sie auch als $\sum_{i=1}^p k_i \cdot s_i = k$ geschrieben.

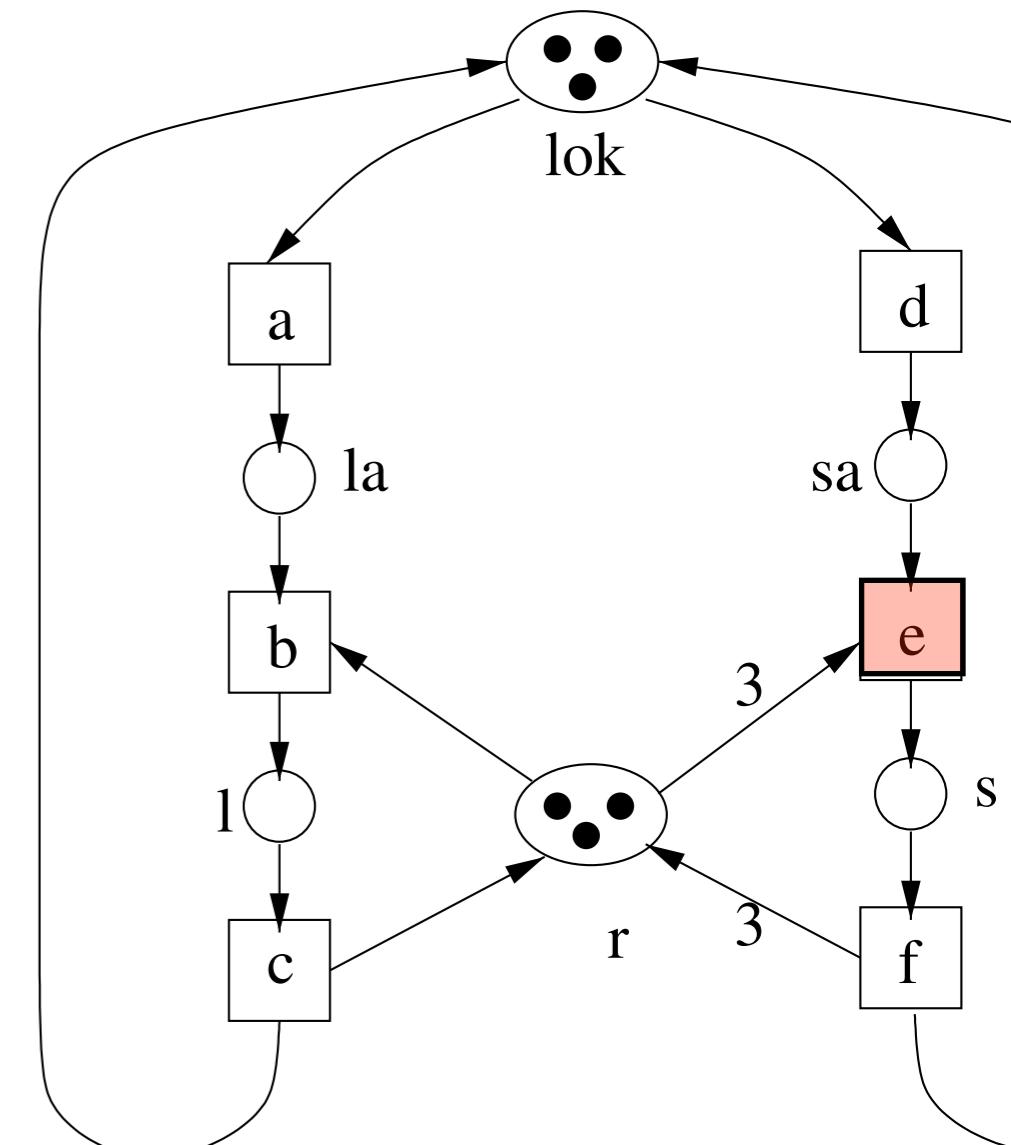
Wirkungsmatrix

Inzidenzmatrix $\Delta_{\mathcal{N}}$

$$\Delta_{\mathcal{N}}(t)(s) = -\tilde{W}(s, t) + \tilde{W}(t, s)$$



\mathcal{N}	a	b	c	d	e	f
lok	-1	0	1	-1	0	1
la	1	-1	0	0	0	0
sa	0	0	0	1	-1	0
l	0	1	-1	0	0	0
s	0	0	0	0	1	-1
r	0	-1	1	0	$-n$	n



$(m_1 \xrightarrow{t} m_2)$, dann gilt:

$$\mathbf{m}_2 = \mathbf{m}_1 + \Delta_{\mathcal{N}}(t)$$

Satz von Lautenbach

Jede ganzzahlige Lösung $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ des linearen Gleichungssystems $\Delta_{\mathcal{N}}^{tr} \cdot i = \underline{0}$ heißt *S-Invarianten-Vektor*⁹ des S/T -Netzes \mathcal{N} .

Satz: [Lautenbach] Wenn $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ ein *S-Invarianten-Vektor* ist, dann gilt $i^{tr} \cdot \mathbf{m} = i^{tr} \cdot \mathbf{m}_0$ für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$.

Beweis:

(durch Induktion über $\mathbf{R}(\mathcal{N})$):

Die Behauptung ist trivial für $\mathbf{m} = \mathbf{m}_0$.

Induktionsschluss:

$(\mathbf{m}_1 \xrightarrow{t} \mathbf{m}_2)$, dann gilt:

$$\mathbf{m}_2 = \mathbf{m}_1 + \Delta_{\mathcal{N}}(t)$$

$$\begin{aligned} i' \cdot \mathbf{m}_2 &= i' \cdot (\mathbf{m}_1 + \Delta_{\mathcal{N}}(t)) \\ &= i' \cdot \mathbf{m}_1 + i' \cdot \Delta_{\mathcal{N}}(t) \\ &= i' \cdot \mathbf{m}_1 \\ &= i' \cdot \mathbf{m}_0 \end{aligned}$$

Beispiel: Berechnung von S-Invarianten

Definition 3.17 Jede ganzzahlige Lösung $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ von $\Delta_{\mathcal{N}}^{tr} \cdot i = \underline{0}$ heißt *S-Invarianten-Vektor⁸* des *S/T-Netzes* \mathcal{N} .

\mathcal{N}	a	b	c	d	e	f
lok	-1	0	1	-1	0	1
la	1	-1	0	0	0	0
sa	0	0	0	1	-1	0
l	0	1	-1	0	0	0
s	0	0	0	0	1	-1
r	0	-1	1	0	$-n$	n

tr

$= 0$

$$\begin{aligned}
 -u + v &= 0 & \Rightarrow u &= v \\
 -v + x - z &= 0 & \Rightarrow x &= u + z \\
 \cancel{-u - x + z = 0} & & & \\
 -u + w &= 0 & \Rightarrow u &= v = w \\
 \cancel{-w + y - n \cdot z = 0} & & & \\
 u - y + n \cdot z &= 0 & \Rightarrow y &= u + n \cdot z
 \end{aligned}$$

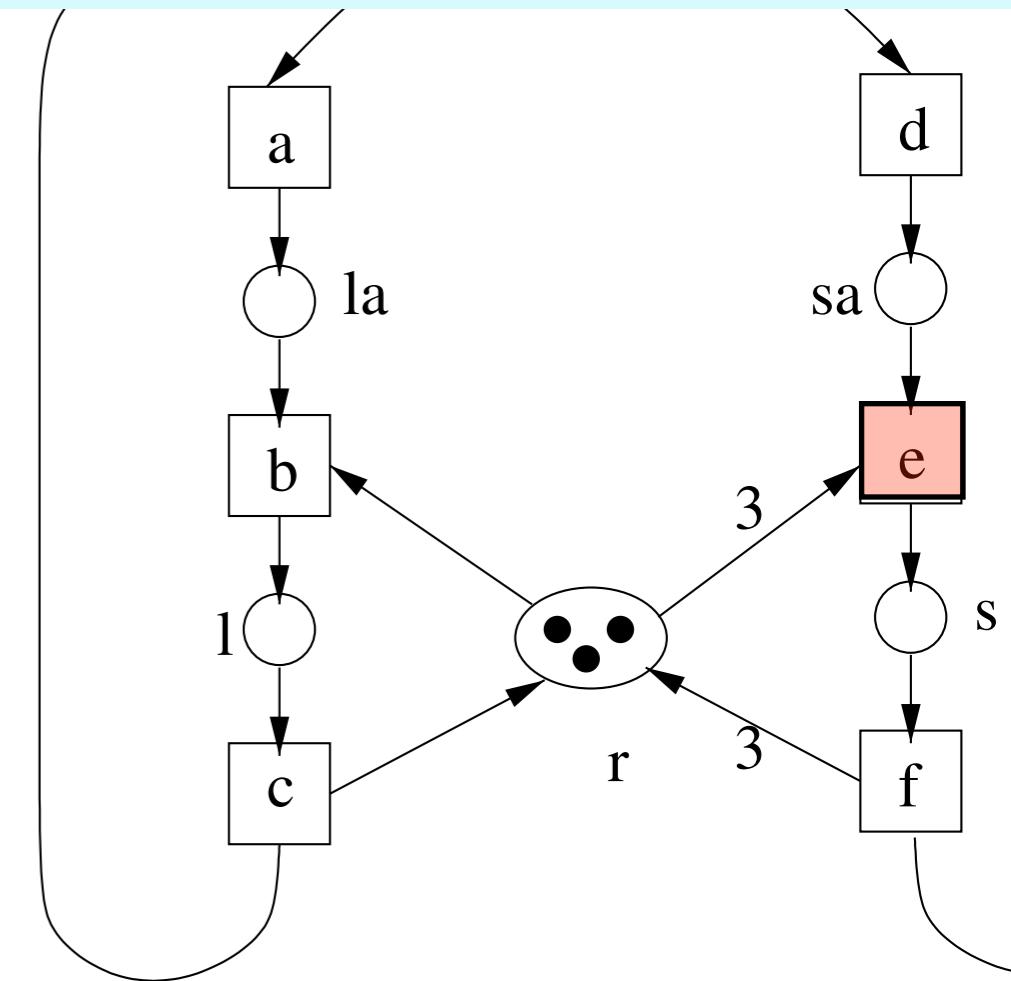
$$\begin{array}{c}
 u \\
 u \\
 u \\
 u+z \\
 u+nz \\
 z
 \end{array}
 = u
 \begin{array}{c}
 1 \\
 1 \\
 1 \\
 1 \\
 1 \\
 0
 \end{array}
 + z
 \begin{array}{c}
 0 \\
 0 \\
 0 \\
 1 \\
 1 \\
 1
 \end{array}$$

- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + n \cdot s = n$

Nachweis von Invarianten

- $i_2 : l + r + n \cdot s = n$

$$\begin{aligned}
 i'_2 \cdot \mathbf{m} &= 1 \cdot \mathbf{m}(l) + n \cdot \mathbf{m}(s) + 1 \cdot \mathbf{m}(r) = \\
 i'_2 \cdot \mathbf{m}_0 &= 1 \cdot \mathbf{m}_0(l) + n \cdot \mathbf{m}_0(s) + 1 \cdot \mathbf{m}_0(r) \\
 &= 1 \cdot 0 + n \cdot 0 + 1 \cdot n = n
 \end{aligned}$$



\mathcal{N}	a	b	c	d	e	f	i_1	i_2
lok	-1	0	1	-1	0	1	1	0
la	1	-1	0	0	0	0	1	0
sa	0	0	0	1	-1	0	1	0
l	0	1	-1	0	0	0	1	1
s	0	0	0	0	1	-1	1	n
r	0	-1	1	0	$-n$	n	0	1

S-Inv.-Vektor vs. S-Inv-Gleichung (Wdh.)

Jede ganzzahlige Lösung $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ des linearen Gleichungssystems $\Delta_{\mathcal{N}}^{tr} \cdot i = \underline{0}$ heißt *S-Invarianten-Vektor*⁹ des *S/T*-Netzes \mathcal{N} .

Satz: [Lautenbach] Wenn $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ ein *S-Invarianten-Vektor* ist, dann gilt $i^{tr} \cdot \mathbf{m} = i^{tr} \cdot \mathbf{m}_0$ für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$.

Definition: Es sei $\mathcal{N} = \langle S, T, F, W, \mathbf{m}_0 \rangle$ ein *S/T*-Netz mit $S = \{s_1, \dots, s_p\}$.

Eine Gleichung der Form $\sum_{i=1}^p k_i \cdot \mathbf{m}(s_i) = k$ mit $k_i, k \in \mathbb{Z}$ heißt *S-Invarianten-Gleichung*, wenn sie für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ gilt.

Abkürzend wird sie auch als $\sum_{i=1}^p k_i \cdot s_i = k$ geschrieben.

S-Inv.-Vektor vs. S-Inv-Gleichung (Wdh.)

Jede ganzzahlige Lösung
heißt *S-Invarianten-Vektor*.

Satz: [Lautenbach] Wenn
 $i^{tr} \cdot \mathbf{m} = i^{tr} \cdot \mathbf{m}_0$ für alle

Also: Aus jedem **S-Invarianten-Vektor** folgt auch sofort eine **Invarianten-Gleichung**.

$$\Delta_{\mathcal{N}}^{tr} \cdot i = 0$$

Umgekehrt gilt dies nicht, denn die Gleichung gilt ja nicht unbedingt **strukturell**, sondern nur für die gegebene **Anfangsmarkierung**.

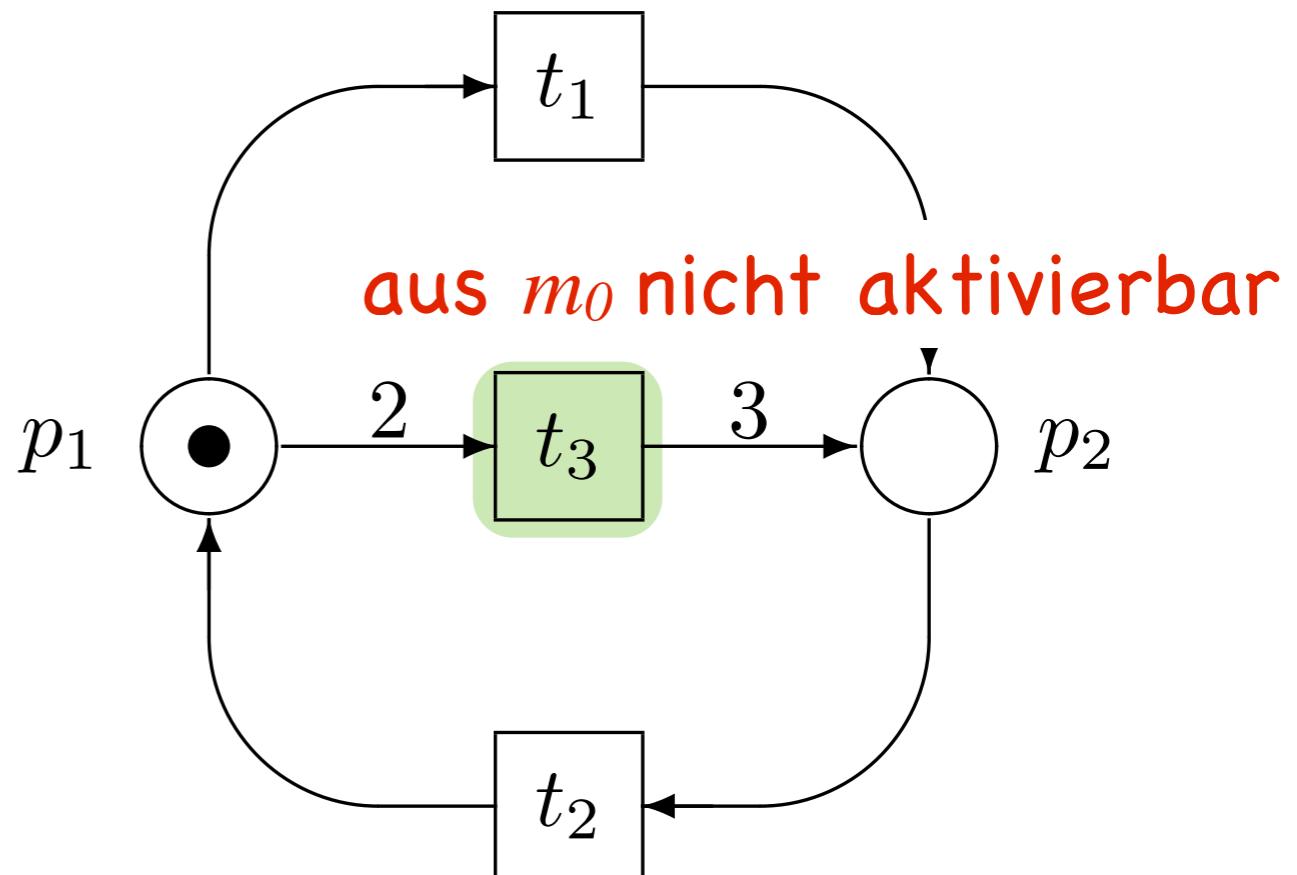
dann gilt

Definition: Es sei $\mathcal{N} = \langle S, T, F, W, \mathbf{m}_0 \rangle$ ein S/T -Netz mit $S = \{s_1, \dots, s_p\}$. Eine Gleichung der Form $\sum_{i=1}^p k_i \cdot \mathbf{m}(s_i) = k$ mit $k_i, k \in \mathbb{Z}$ heißt *S-Invarianten-Gleichung*, wenn sie für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ gilt. Abkürzend wird sie auch als $\sum_{i=1}^p k_i \cdot s_i = k$ geschrieben.

Strukturelle Invarianten vs. Inv.-Gleichung

Also: Aus jedem S-Invarianten-Vektor folgt auch sofort eine Invarianten-Gleichung.

Umgekehrt gilt dies nicht, denn die Gleichung gilt ja nicht unbedingt **strukturell**, sondern nur für die gegebene Anfangsmarkierung.



$$\Delta = \begin{pmatrix} -1 & 1 & -2 \\ 1 & -1 & 3 \end{pmatrix}$$

Es gilt die Invarianten-Gleichung:

$$\forall \mathbf{m} \in \mathbf{R}(N): 1 \cdot \mathbf{m}(p_1) + 1 \cdot \mathbf{m}(p_2) = 1 \cdot \mathbf{m}_0(p_1) + 1 \cdot \mathbf{m}_0(p_2) = 1$$

Der zugehörige Vektor $i = (1, 1)^{tr}$ ist jedoch kein Invariantenvektor, denn

$$\Delta^{tr} i = \begin{pmatrix} -1 & 1 \\ 1 & -1 \\ -2 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Strukturelle Beschränktheit

Theorem: Besitzt das P/T Netz \mathcal{N} eine überdeckende, positive P -Invariante \mathbf{i} , d.h. $\mathbf{i}(p) > 0$ für alle p , dann ist \mathcal{N} strukturell beschränkt.

Beweis: Sei \mathbf{m}_0 eine beliebige Initialmarkierung. Mit dem Satz von Lautenbach gilt für jede erreichbare Markierung \mathbf{m} :

$$\mathbf{i} \cdot \mathbf{m} = \sum_{p \in P} \mathbf{i}(p) \cdot \mathbf{m}(p) = \sum_{p \in P} \mathbf{i}(p) \cdot \mathbf{m}_0(p)$$

Da $\mathbf{i}(p) > 0$ für jede Stelle gilt, erhalten wir:

$$\mathbf{i}(p) \cdot \mathbf{m}(p) \leq \sum_{p \in P} \mathbf{i}(p) \cdot \mathbf{m}(p)$$

Also gilt für p die Beschränkung (wieder mit $\mathbf{i}(p) > 0$):

$$\mathbf{m}(p) \leq \frac{(\mathbf{i} \cdot \mathbf{m}_0)}{\mathbf{i}(p)} \leq \mathbf{i} \cdot \mathbf{m}_0$$

Da für jedes Netz die Anfangsmarkierung fest ist, existiert also für jede Anfangsmarkierung und jeden Platz eine obere Schranke für die Markenzahl. \square

T-Invarianten

Definition 3.17 Jede ganzzahlige Lösung $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ von $\Delta_{\mathcal{N}}^{tr} \cdot i = \underline{0}$ heißt S -Invarianten-Vektor⁸ des S/T -Netzes \mathcal{N} .

Definition 5.20 Jede Lösung $j \in \mathbb{N}^{|T|} \setminus \{\underline{0}\}$ von $\Delta_{\mathcal{N}} \cdot j = \underline{0}$ heißt T-Invarianten-Vektor des S/T -Netzes \mathcal{N} .

$$\mathbf{m}_1 \xrightarrow{w} \mathbf{m}_2. \quad w = t_{i_1} \dots t_{i_k}$$

$$\mathbf{m}_2 = \mathbf{m}_1 + \underbrace{\Delta_{\mathcal{N}}(t_{i_1}) + \Delta_{\mathcal{N}}(t_{i_2}) + \dots + \Delta_{\mathcal{N}}(t_{i_k})}_{\text{Wann ist dieser Teil der Nullvektor?}}$$

Antwort:

Falls $\Delta_{\mathcal{N}} \cdot j = \underline{0}$

$$j = \psi(w) := (|w|_{x_1}, |w|_{x_2}, \dots, |w|_{x_k})$$

Definition 3.17 Jede ganzzahlige Lösung $i \in \mathbb{Z}^{|S|} \setminus \{\underline{0}\}$ von $\Delta_{\mathcal{N}}^{tr} \cdot i = \underline{0}$ heißt **S-Invarianten-Vektor⁸** des S/T -Netzes \mathcal{N} .

\mathcal{N}	a	b	c	d	e	f
lok	-1	0	1	-1	0	1
la	1	-1	0	0	0	0
sa	0	0	0	1	-1	0
l	0	1	-1	0	0	0
s	0	0	0	0	1	-1
r	0	-1	1	0	$-n$	n

tr
 $\begin{matrix} u \\ v \\ w \\ x \\ y \\ z \end{matrix}$
 $= 0$

$$\begin{aligned} -u + v &= 0 & \Rightarrow u &= v \\ -v + x - z &= 0 & \Rightarrow x &= u + z \\ \cancel{-u - x + z = 0} & & & \\ -u + w &= 0 & \Rightarrow u &= v = w \end{aligned}$$
 $\begin{matrix} u \\ u \\ u \\ u+z \\ u+nz \\ z \end{matrix}$
 $= u$
 $\begin{matrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{matrix}$
 $+z$
 $\begin{matrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{matrix}$

$$\begin{aligned} \cancel{w + y - n \cdot z = 0} \\ u - y + n \cdot z = 0 \end{aligned} \Rightarrow y = u + n \cdot z$$

- $i_1 : lok + la + sa + l + s = n$
- $i_2 : l + r + n \cdot s = n$

Reproduktion von Markierungen

Satz: Es seien $\mathbf{m}_1, \mathbf{m}_2$ Markierungen und $w = t_{i_1} \dots t_{i_k}$ eine Schaltfolge, die \mathbf{m}_1 in \mathbf{m}_2 überführt: $\mathbf{m}_1 \xrightarrow{w} \mathbf{m}_2$.

Die Markierungen \mathbf{m}_1 und \mathbf{m}_2 sind genau dann gleich, wenn es einen T -Invarianten-Vektor $j \in \mathbb{N}^{|T|}$ derart gibt, dass jede Transition $t \in T$ genau $j(t)$ mal in w vorkommt, d.h.: $j = \psi(w)$.

Beweis: Es gilt nach Voraussetzung

$$\mathbf{m}_1 = \mathbf{m}_2 = \mathbf{m}_1 + \Delta_{\mathcal{N}}(t_{i_1}) + \Delta_{\mathcal{N}}(t_{i_2}) + \dots + \Delta_{\mathcal{N}}(t_{i_k})$$

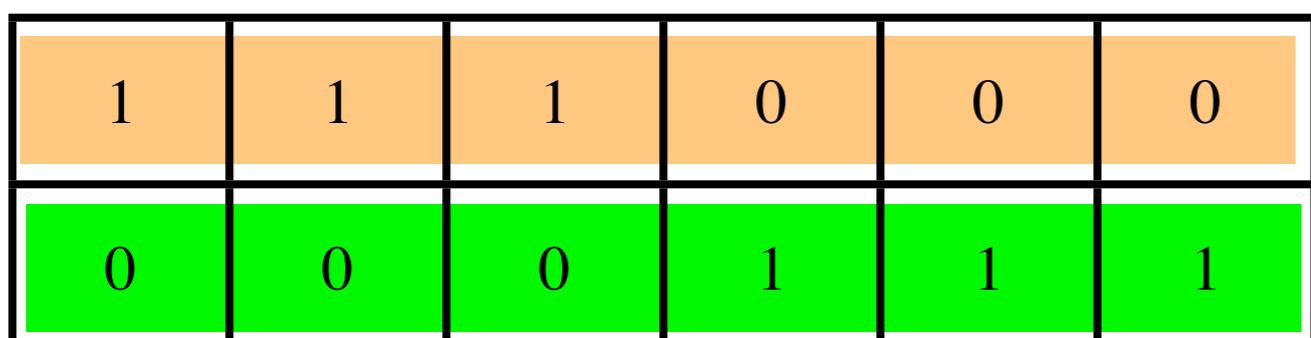
genau dann, wenn gilt:

$$\begin{aligned} 0 &= \Delta_{\mathcal{N}}(t_{i_1}) + \dots + \Delta_{\mathcal{N}}(t_{i_k}) \\ &= \Delta_{\mathcal{N}} \cdot e_{t_{i_1}} + \dots + \Delta_{\mathcal{N}} \cdot e_{t_{i_k}} \\ &= \Delta_{\mathcal{N}} \cdot (|w|_{t_1} e_{t_1}) + \dots + \Delta_{\mathcal{N}} \cdot (|w|_{t_{|T|}} e_{t_{|T|}}) \\ &= \Delta_{\mathcal{N}} \cdot \psi(w) \end{aligned}$$

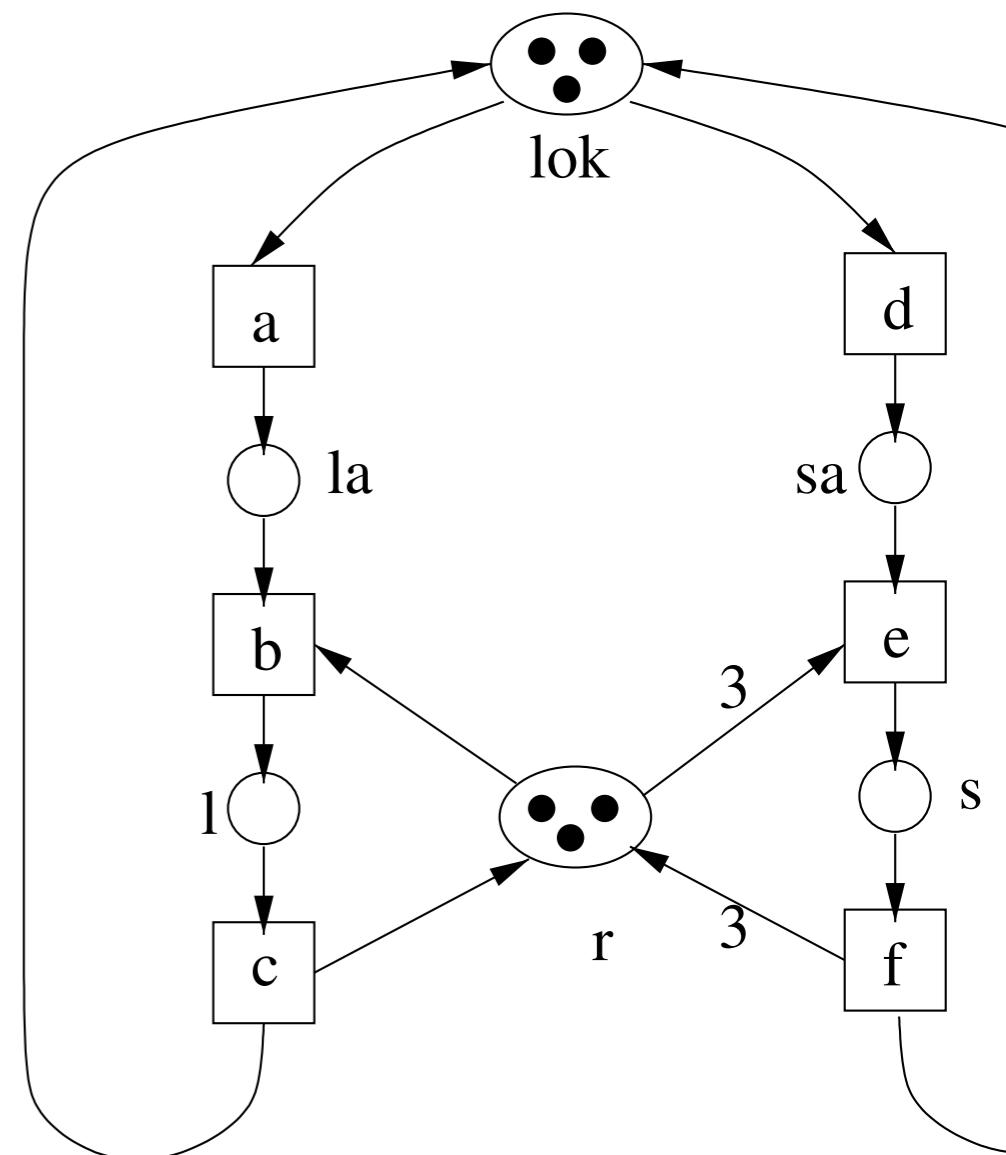
Mit anderen Worten: $\psi(w)$ ist eine T -Invariante. □

Beispiel

\mathcal{N}	a	b	c	d	e	f
lok	-1	0	1	-1	0	1
la	1	-1	0	0	0	0
sa	0	0	0	1	-1	0
l	0	1	-1	0	0	0
s	0	0	0	0	1	-1
r	0	-1	1	0	$-n$	n



j	3	3	3	2	2	2
-----	---	---	---	---	---	---



Theorem: Sei \mathcal{N} ein beschränktes P/T-Netz, das in \mathbf{m} die unendliche Schaltfolge $w \in T^\omega$ aktiviert. Dann existiert eine T -Invariante j mit $j(t) = 0$, falls t nicht oder nur endlich oft in w vorkommt $j(t) > 0$, falls t unendlich oft in w vorkommt.

Folgendes Theorem erlaubt es, für beschränkte Netze durch Invariantenanalyse Lebendigkeit zu verwerfen: Besitzt ein beschränktes Netz keine T -Invariante, dann kann es nicht lebendig sein.

Theorem: Sei \mathcal{N} ein beschränktes und lebendiges P/T-Netz. Dann existiert eine echt positive T -Invariante j , d.h. $j(t) > 0$ für alle $t \in T$.

Beweis: Da das Netz lebendig ist, existiert eine unendliche Schaltfolge w , in der alle Transitionen unendlich oft vorkommen. Mit Theorem 1.7.2 existiert dann eine echt positive T -Invariante.

Man kann dies auch direkt beweisen: Wenn \mathcal{N} lebendig ist, dann gibt es zur Initialmarkierung \mathbf{m}_0 eine Schaltfolge, in der alle Transitionen unendlich oft vorkommen. Da das Netz beschränkt ist, kommt in dieser Folge $\mathbf{m}_0 \xrightarrow{t_{i_1}} \mathbf{m}_1 \xrightarrow{t_{i_2}} \mathbf{m}_2 \dots$ mindestens eine Markierung doppelt vor: $\mathbf{m}_i \xrightarrow{w} \mathbf{m}_j$ mit $\mathbf{m}_i = \mathbf{m}_j$. Also ist nach Satz 1.7.2 $\Psi(w)$ eine T -Invariante. \square

Fallen

Fallen Wir betrachten Stellenmengen $A \subseteq P$ mit der Eigenschaft, dass alle Transition t , die aus A Marken entfernen, auch wieder Marken in A generieren.

Definition: Eine Stellenmenge $A \subseteq P$ heißt *Falle* (engl. *trap*), wenn jede Transition t , die aus A Marken entfernt, auch wieder Marken in A generiert:

$$\forall t \in T : t \in A^\bullet \Rightarrow t \in {}^\bullet A \quad \text{bzw. kurz} \quad A^\bullet \subseteq {}^\bullet A$$

Eine Falle $A \subseteq P$ heißt in m markiert, wenn mindestens eine Stelle in A markiert ist:

$$\sum_{p \in A} \mathbf{m}(p) > 0$$

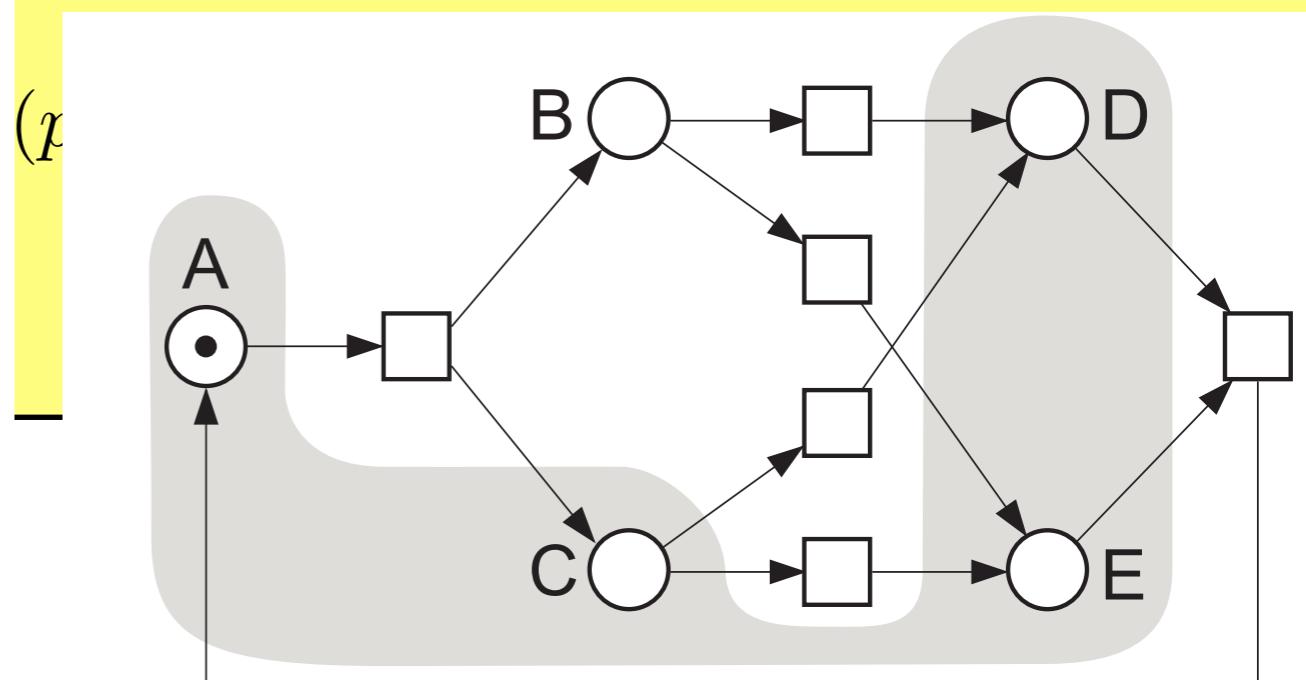
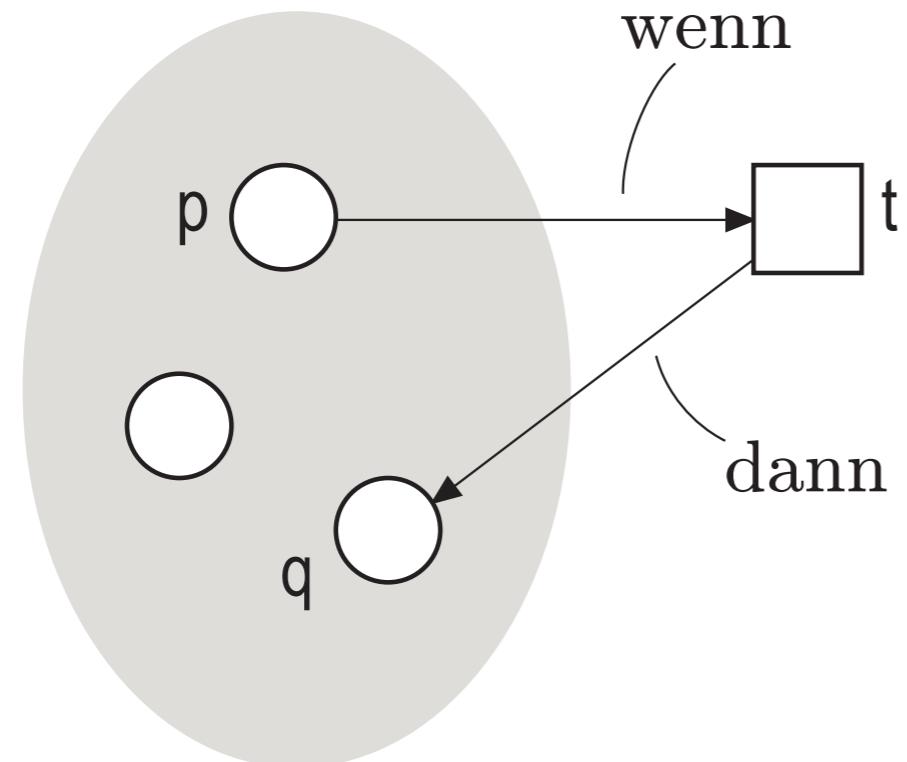
Fallen

Fallen Wir betrachten Stellenmengen $A \subseteq P$ mit der Eigenschaft, dass alle Transition t , die aus A Marken entfernen, auch wieder Marken in A generieren.

Definition: Eine Stellenmenge $A \subseteq P$ heißt *Falle* (engl. *trap*), wenn jede Transition t , die aus A Marken entfernt, auch wieder Marken in A generiert:

$$\forall t \in T : t \in A^\bullet \Rightarrow t \in {}^\bullet A \quad \text{bzw. kurz} \quad A^\bullet \subseteq {}^\bullet A$$

Eine Falle $A \subseteq P$ heißt in m markiert, wenn mindestens eine Stelle in A in m



Falle: {A,C,D,E}

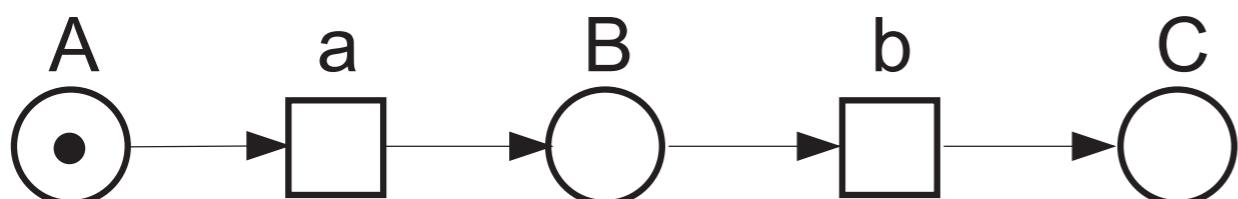
Falle {p, q, ...}

Marken in der Falle

Da jede Transition t , die Marken aus der Falle entfernt, auch wieder welche in ihr erzeugt, ist es nicht möglich, eine markierte Falle komplett zu leeren. Es ist also mindestens eine Marke in der Falle gefangen.

Lemma: Sei A eine Falle eines einfachen P/T-Netzes \mathcal{N} . Ist die Falle A in einer Markierung \mathbf{m}_0 markiert, dann bleibt sie dies auch in allen von \mathbf{m}_0 aus erreichbaren Markierungen.

$$\forall \mathbf{m}_0 : \sum_{p \in A} \mathbf{m}_0(p) > 0 \Rightarrow \forall \mathbf{m} \in R(N, \mathbf{m}_0) : \sum_{p \in A} \mathbf{m}(p) > 0$$



markierte Falle: {A,B,C}

unmarkierte Fallen: {B,C}, {C}

Co-Fallen/Siphon/Abfluss

Siphons/Co-Fallen Die Umkehrung des Konzeptes *Falle* existiert auch. Es wird als *Siphon* oder auch als *co-Falle* bezeichnet.

Definition: Eine Stellenmenge $A \subseteq P$ heißt *Siphon*, wenn jede Transition t , die Marken in A generiert, dazu welche aus A benötigt:

$$\forall t \in T : t \in {}^\bullet A \Rightarrow t \in A^\bullet \quad \text{bzw. kurz} \quad {}^\bullet A \subseteq A^\bullet$$

Ein Siphon $A \subseteq P$ heißt *markiert*, wenn mindestens eine Stelle in A markiert ist.

$$\sum_{p \in A} \mathbf{m}(p) > 0$$

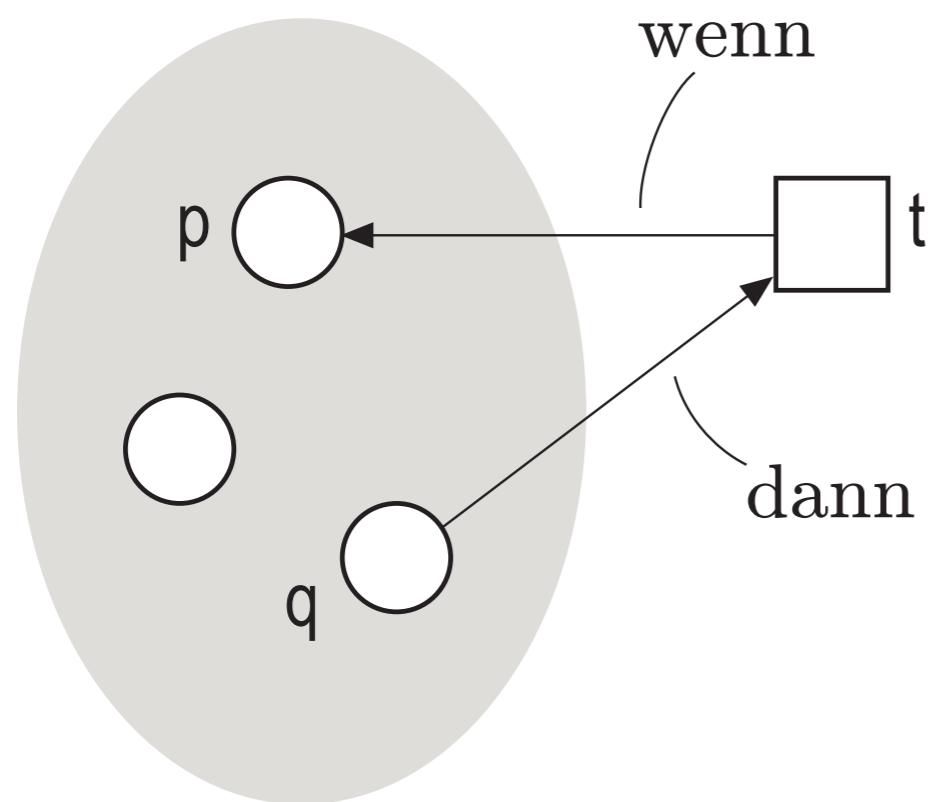
Bemerkung: Nach Definition ist die leere Menge sowohl ein Siphon als auch eine Trap. Ebenso ist die Stellenmenge P selbst sowohl ein Siphon als auch eine Trap.

Co-Fallen/Siphon/Abfluss

Siphons/Co-Fallen Die Umkehrung des Konzeptes *Falle* existiert auch. Es wird als *Siphon* oder auch als *co-Falle* bezeichnet.

Definition: Eine Stellenmenge $A \subseteq P$ heißt *Siphon*, wenn jede Transition t , die Marken in A generiert, dazu welche aus A benötigt:

$$\forall t \in T : t \in {}^\bullet A \Rightarrow t \in A^\bullet \quad \text{bzw. kurz} \quad {}^\bullet A \subseteq A^\bullet$$

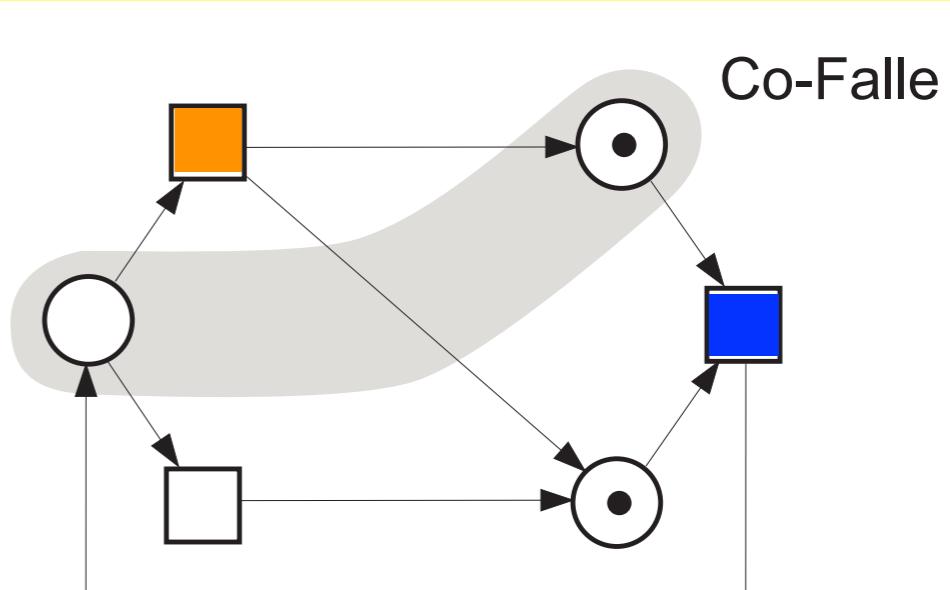


Co-Falle $\{p, q, \dots\}$

enn mindestens eine Stelle in A markiert

$$n(p) > 0$$

eere
ge F

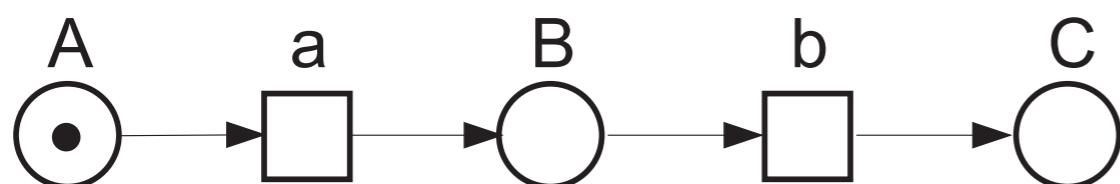


Ein “ausgesprudelter” Siphon

Da ein Siphon nur durch Transition t gefüllt werden, die zum Schalten Marken aus dem Siphon benötigen, muss ein unmarkierter Siphon dies auch in allen erreichbaren Markierungen bleiben.

Lemma: Sei A ein Siphon eines einfachen P/T-Netzes \mathcal{N} . Ist der Siphon A in einer Markierung \mathbf{m}_0 unmarkiert, dann bleibt er dies auch in allen von \mathbf{m}_0 aus erreichbaren Markierungen.

$$\forall \mathbf{m}_0 : \sum_{p \in A} \mathbf{m}_0(p) = 0 \Rightarrow \forall \mathbf{m} \in R(N, \mathbf{m}_0) : \sum_{p \in A} \mathbf{m}(p) = 0$$



Co-Fallen: {A,B,C}, {A,B}, {A}

Strukturelle Nicht-Lebendigkeit

Aus strukturellen Eigenschaften lassen sich dynamische folgern. Mit dem folgenden Lemma können wir insbesondere die nachweisen, dass ein Netz nicht lebendig sein kann.

Lemma: Sei \mathcal{N} ein einfaches P/T-Netz ohne isolierte Plätze. Wenn \mathcal{N} ein lebendiges Netz ist, dann muss jeder nichtleere Siphon A in der Initialmarkierung markiert sein.

Beweis: Wäre der Siphon A in der Initialmarkierung unmarkiert, dann wären alle Transitionen aus A^\bullet tot und es gilt $A^\bullet \neq \emptyset$, da \mathcal{N} keine isolierte Plätze besitzt. Also existiert mindestens eine tote Transition. Widerspruch. \square

Siphon/Trap-Eigenschaft Wir können beide Eigenschaften – Falle und Co-Falle – sinnvoll kombinieren: Eine markierte Falle A wird nie leer. Ist A Teilmenge eines Siphons B , dann kann auch B nie leer werden.

Definition: Ein Systemnetz \mathcal{N} hat die *Siphon/Trap-Eigenschaft*, wenn gilt: Jeder Siphon B von \mathcal{N} enthält eine anfangsmarkierte Falle A als Teilmenge.

Angenommen, \mathcal{N} besitzt die Siphon/Trap-Eigenschaft, dann aktiviert jede erreichbare Markierung mindestens eine Transition.

Satz: Wenn \mathcal{N} die Siphon/Trap-Eigenschaft besitzt, dann aktiviert jede erreichbare Markierung von \mathcal{N} mindestens eine Transition, d.h. \mathcal{N} ist verklemmungsfrei.

Beweis: Angenommen \mathcal{N} wäre nicht verklemmungsfrei. Wenn \mathbf{m} eine Verklemmung ist, dann ist die Menge $B := \{p \mid \mathbf{m}(p) = 0\}$ ein Siphon: Sei $t \in {}^\bullet B$, dann muss es mindestens einen Platz $p \in {}^\bullet t$ geben, der auch in B liegt, denn andernfalls wäre t ja wegen $W(x, y) \leq 1$ aktiviert und \mathbf{m} wäre keine Verklemmung. Also gilt $t \in p^\bullet \subseteq B^\bullet$, also gilt ${}^\bullet B \subseteq B^\bullet$, d.h. B ist ein Siphon.

Da B unmarkiert ist, enthält B erst recht keine anfangsmarkierte Falle als Teilmenge. Also besitzt \mathcal{N} nicht die Siphon/Trap-Eigenschaft. Widerspruch. □

FGI 2

Daniel Moldt

Turing-Mächtigkeit und Inhibitornetze

Turing-Mächtigkeit von Netzen?

Turing-Maschine



(2-)Zählerautomat

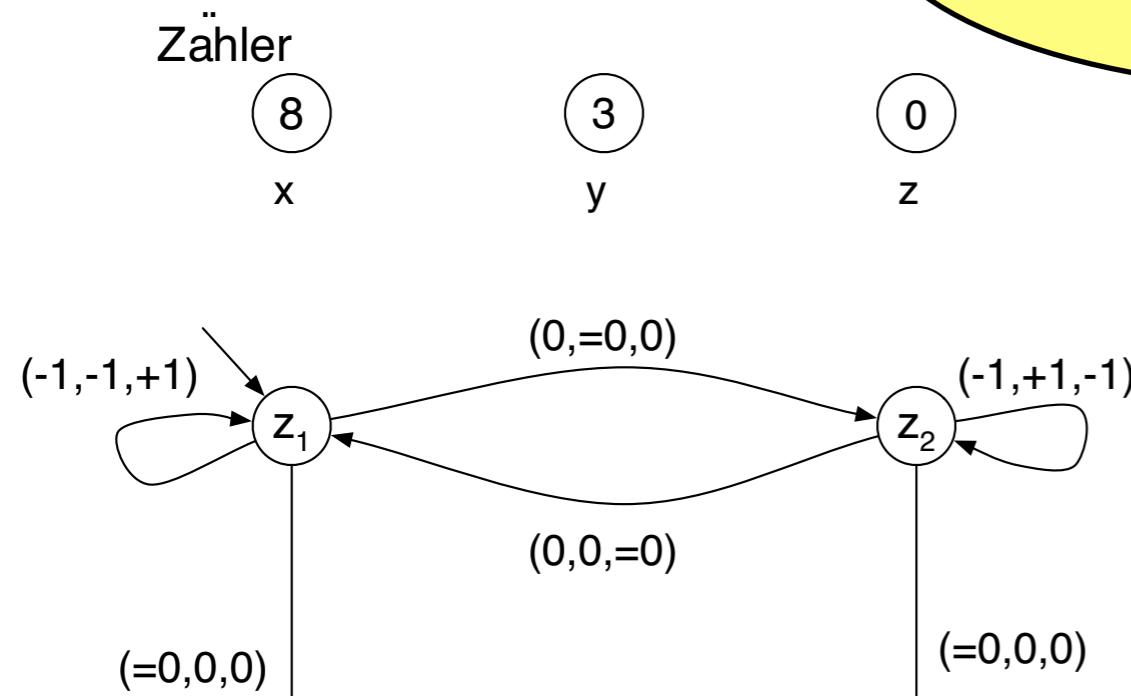


P/T-Netz

Zählerautomaten

Hier: 3 Zähler

$$x := x \bmod y$$



erhöhe Zähler #1

$(+1, -1, 0)$

belasse Zähler #3

erniedrige Zähler #2
falls größer Null

Übergang nur
falls Zähler #3
gleich 0

$(+1, -1, =0)$

Zähler

$x := x \bmod y$

8

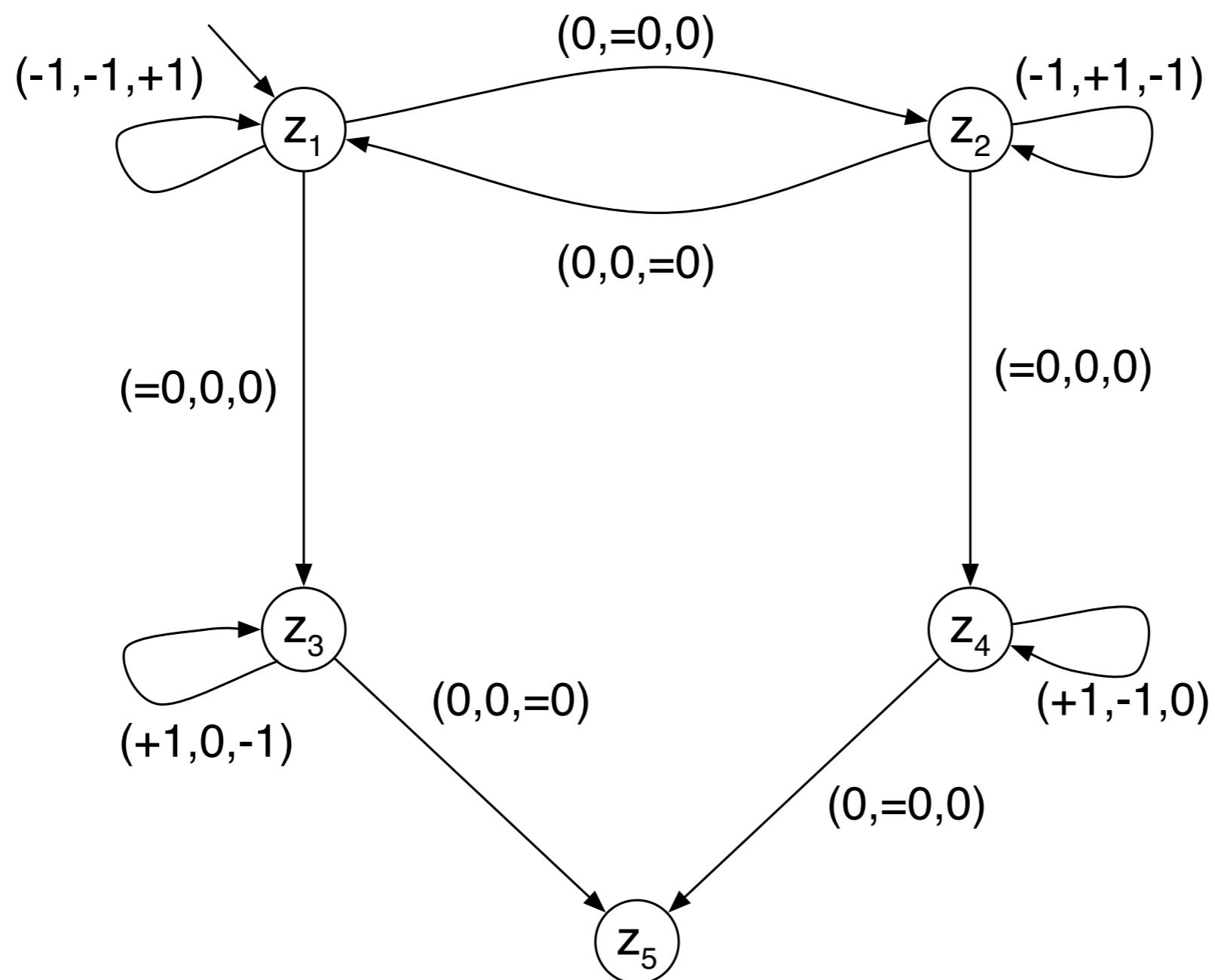
x

3

y

0

z



Zähler 7 2 1

8

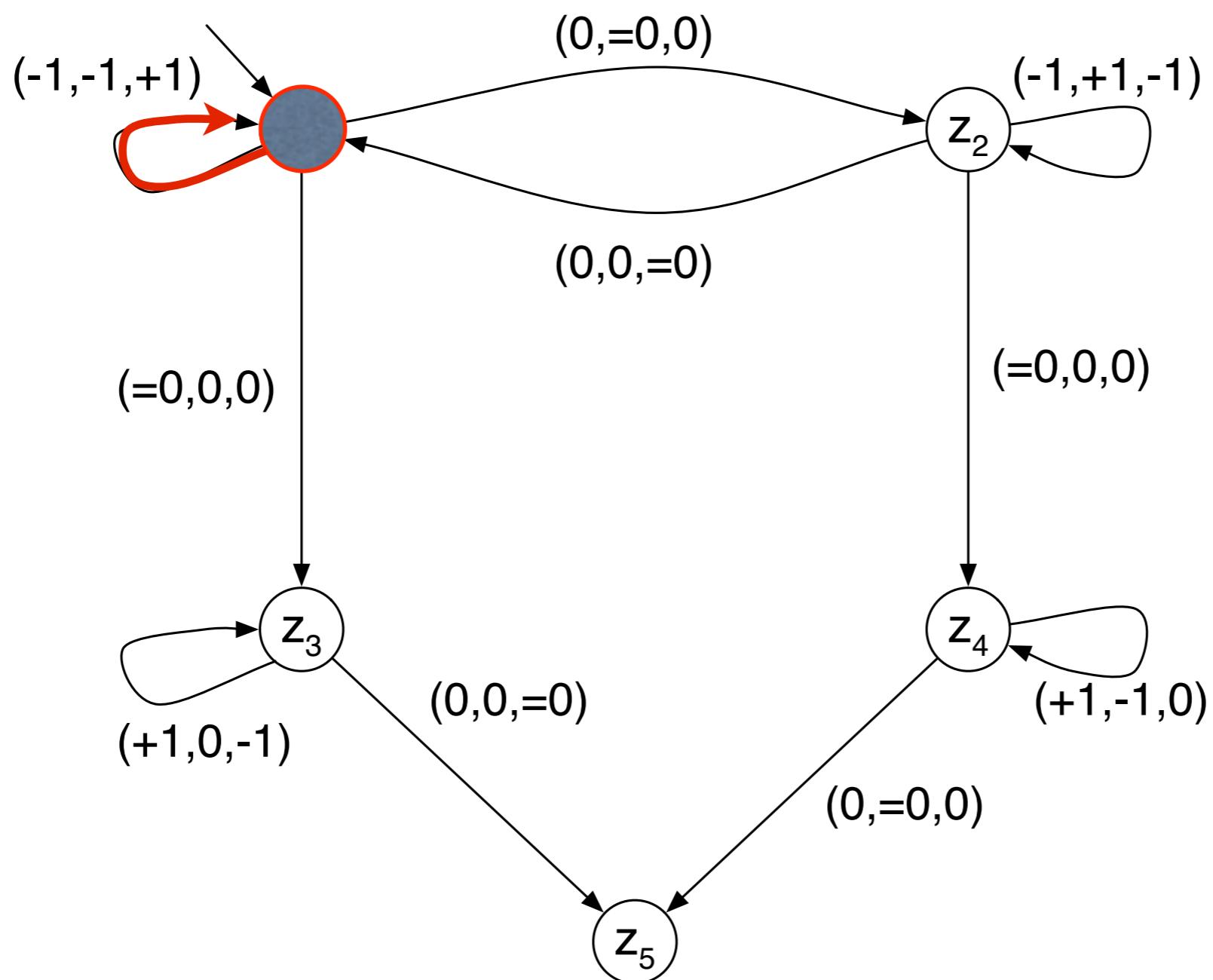
x

3

y

0

z



Zähler $\begin{matrix} 7 \\ 6 \end{matrix}$

8

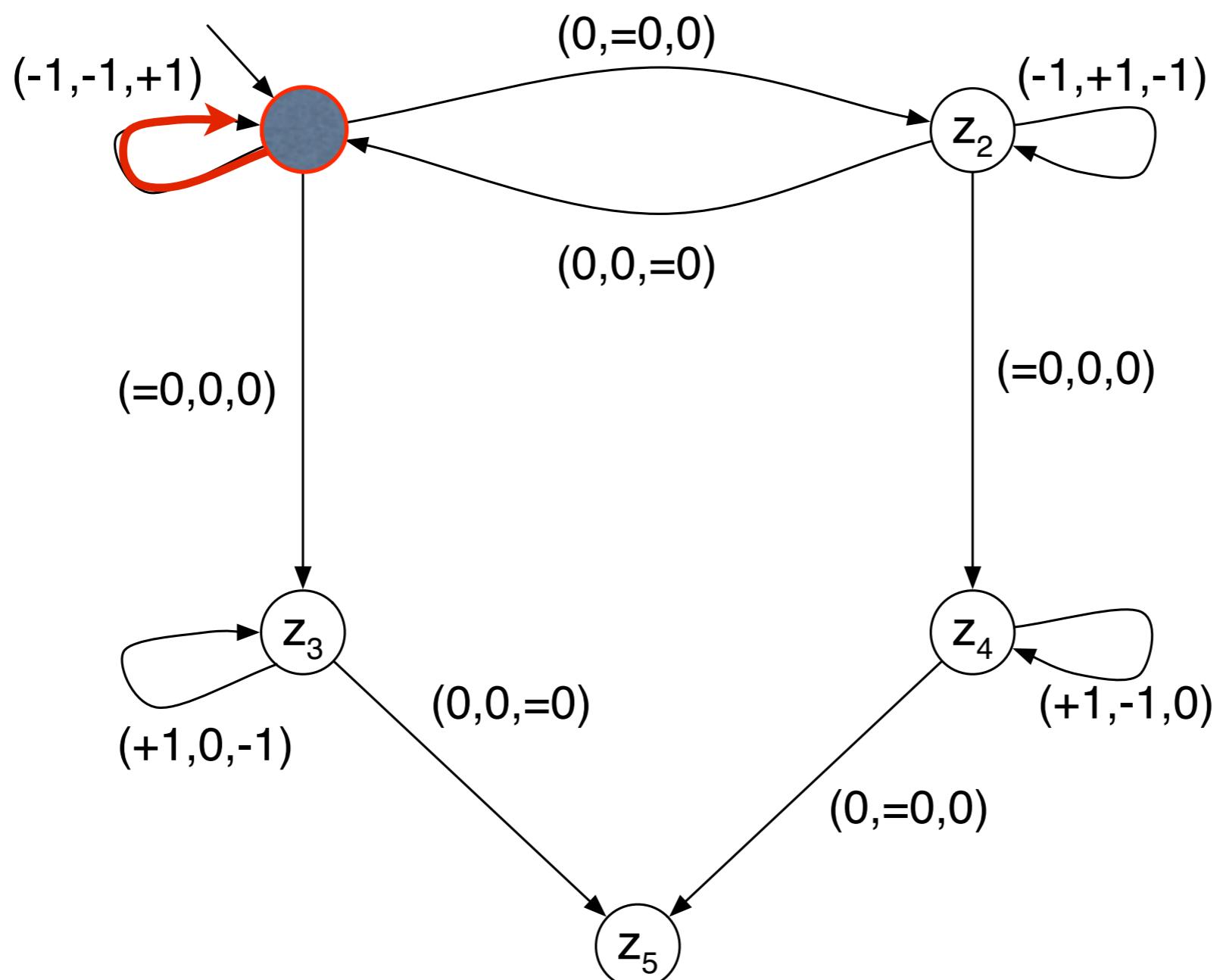
x

3

y

0

z



Zähler $\begin{matrix} 7 \\ 6 \\ 5 \end{matrix}$

8

x

$\begin{matrix} 2 \\ 1 \\ 0 \end{matrix}$

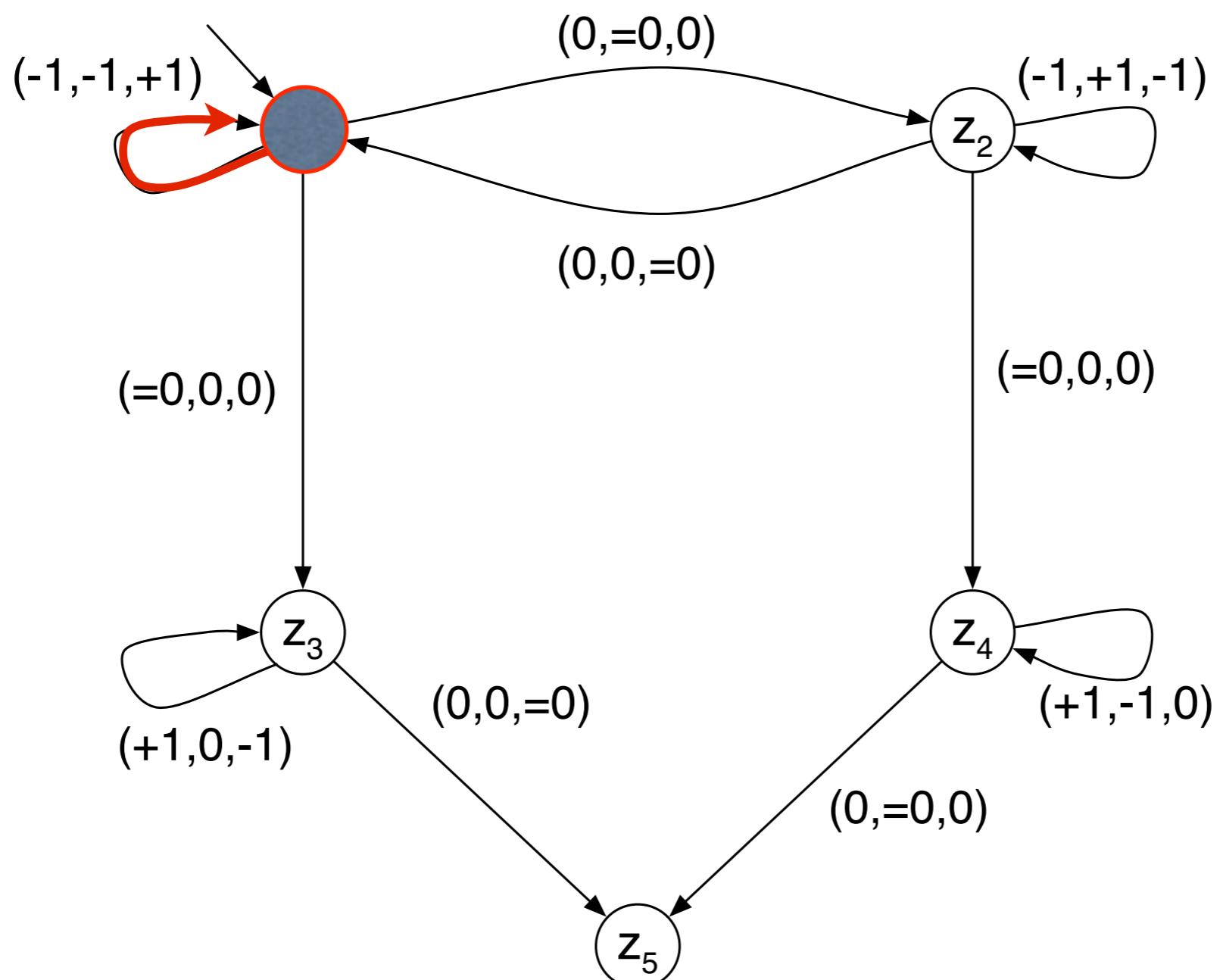
3

y

$\begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$

0

z



Zähler $\begin{matrix} 7 \\ 6 \\ 5 \end{matrix}$

8

x

$\begin{matrix} 2 \\ 1 \\ 0 \end{matrix}$

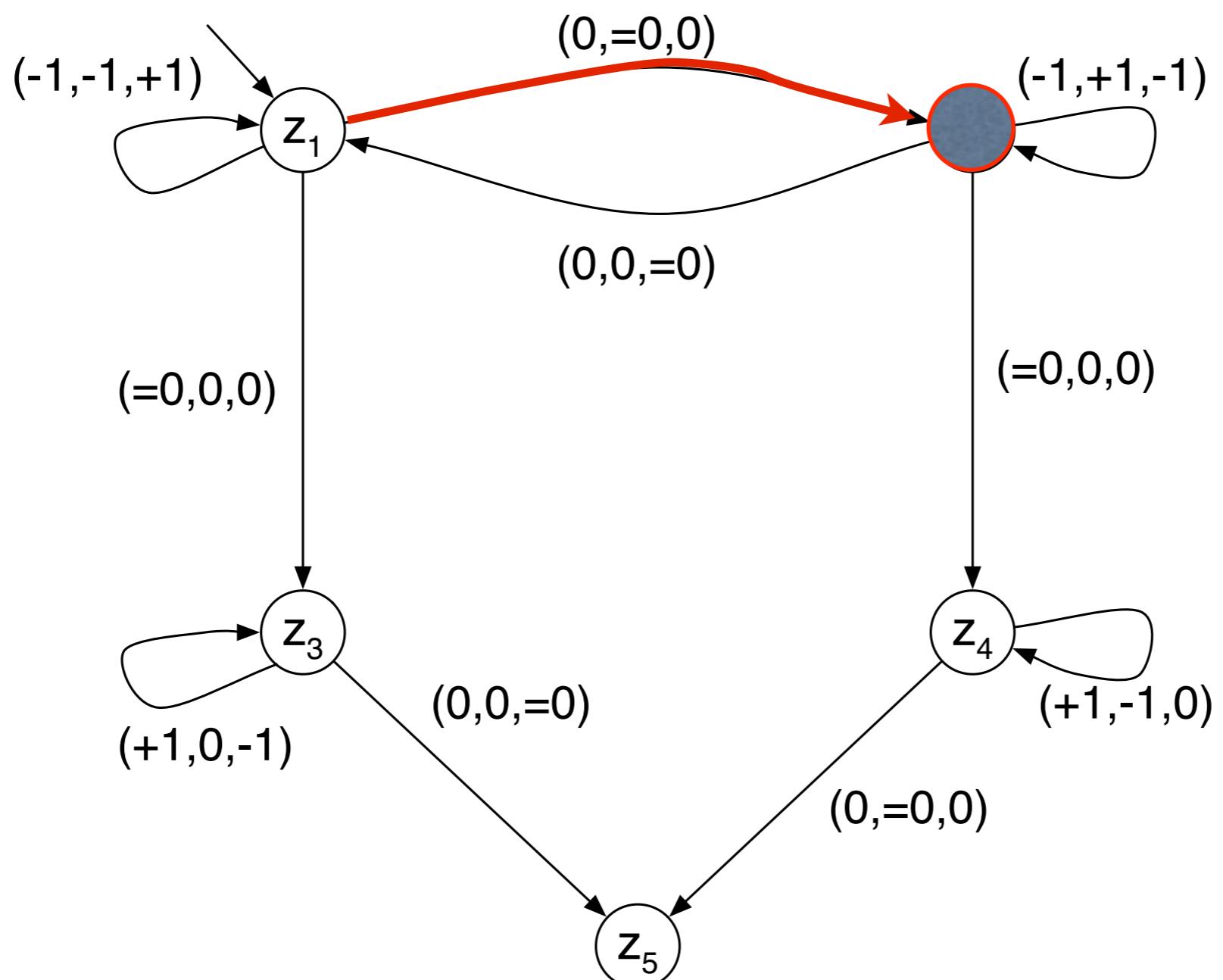
3

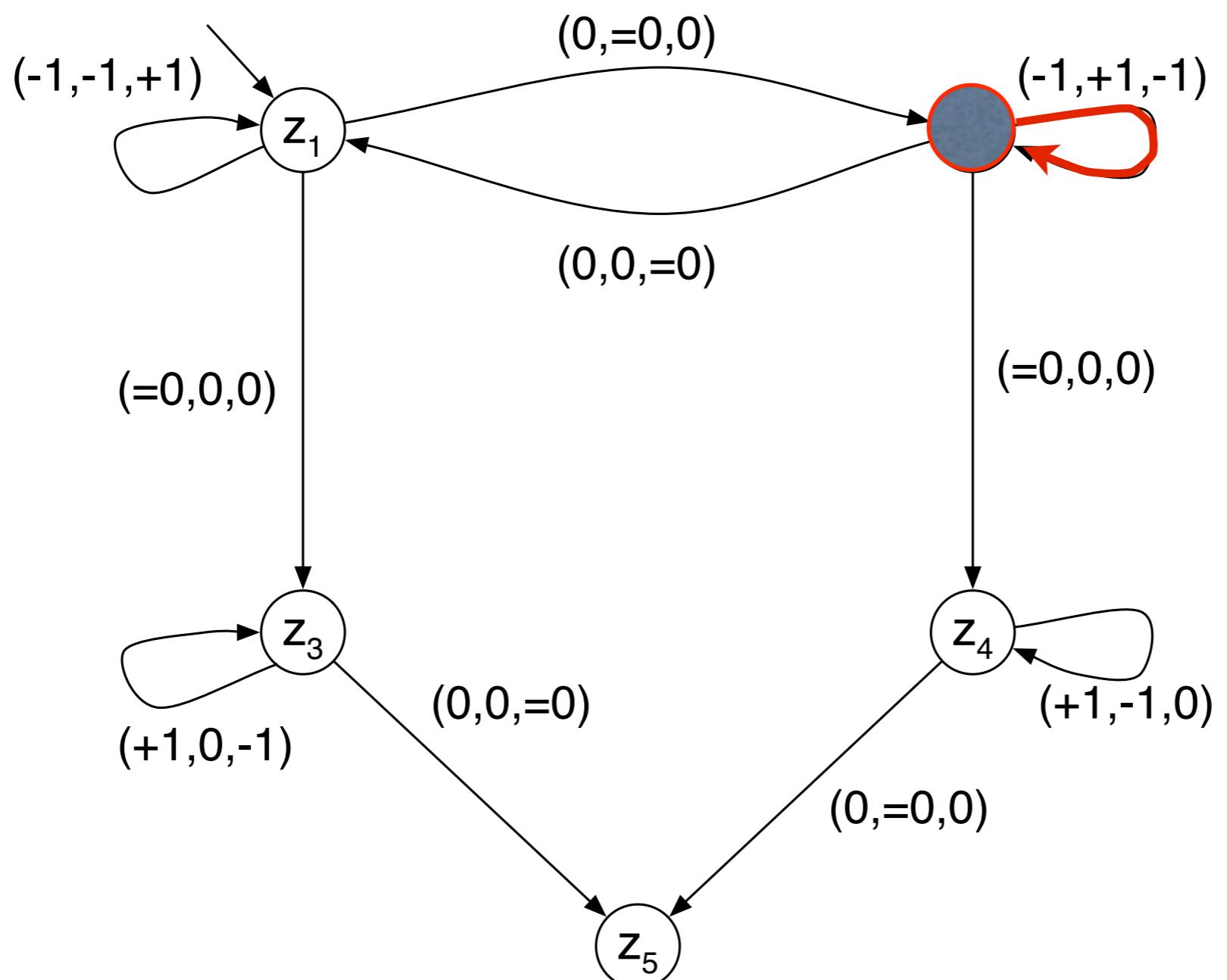
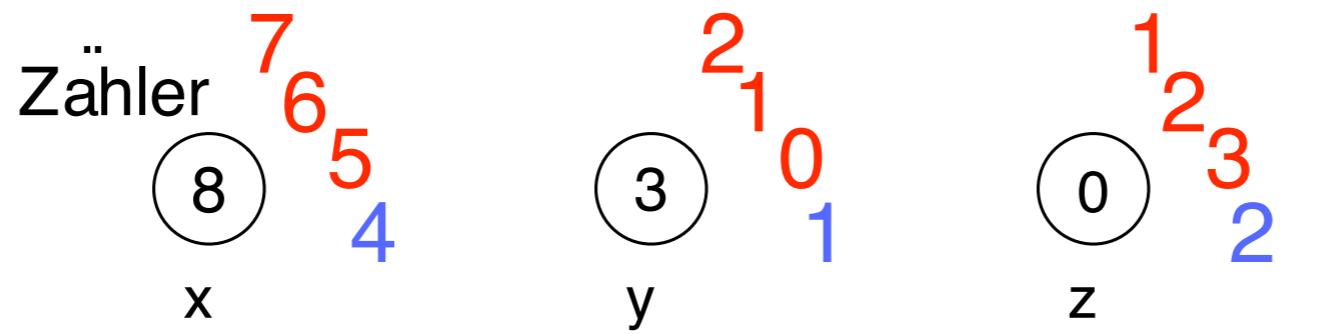
y

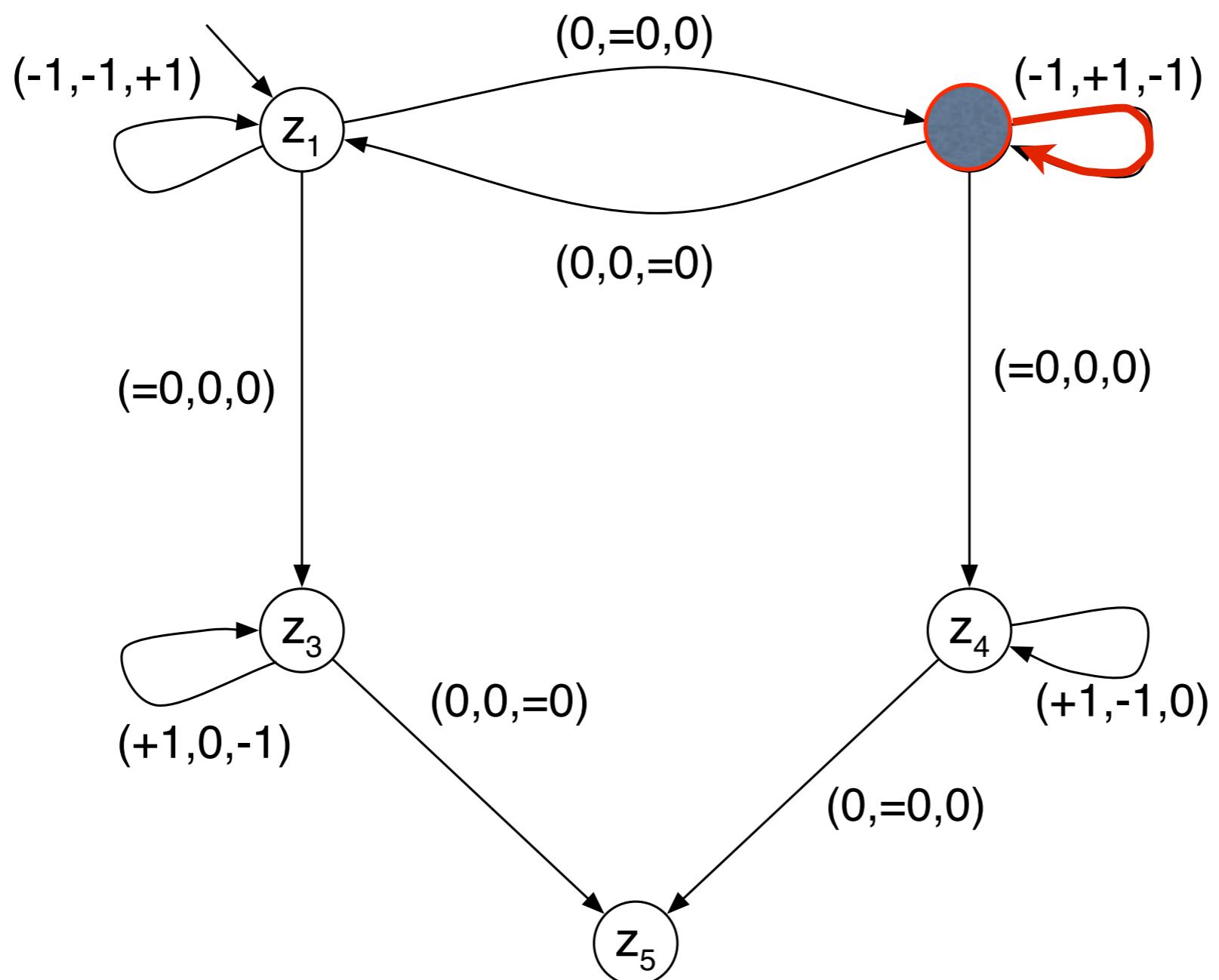
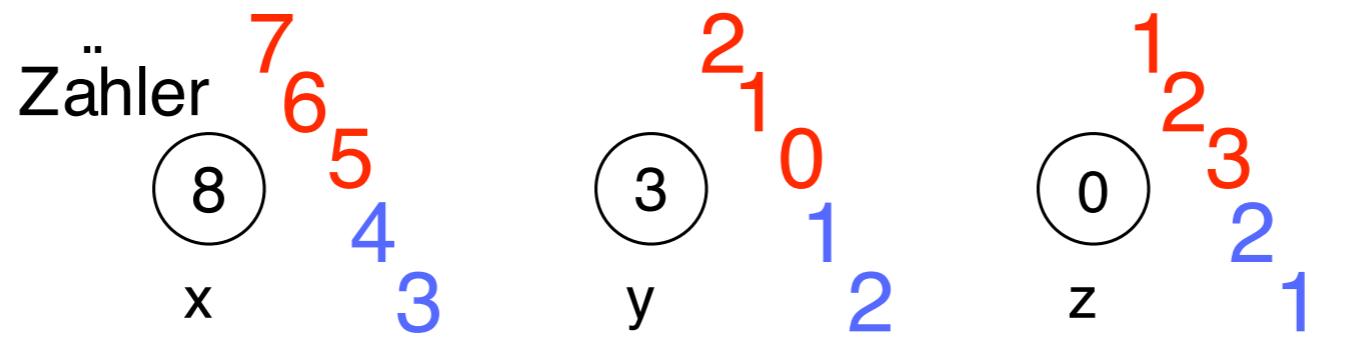
$\begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$

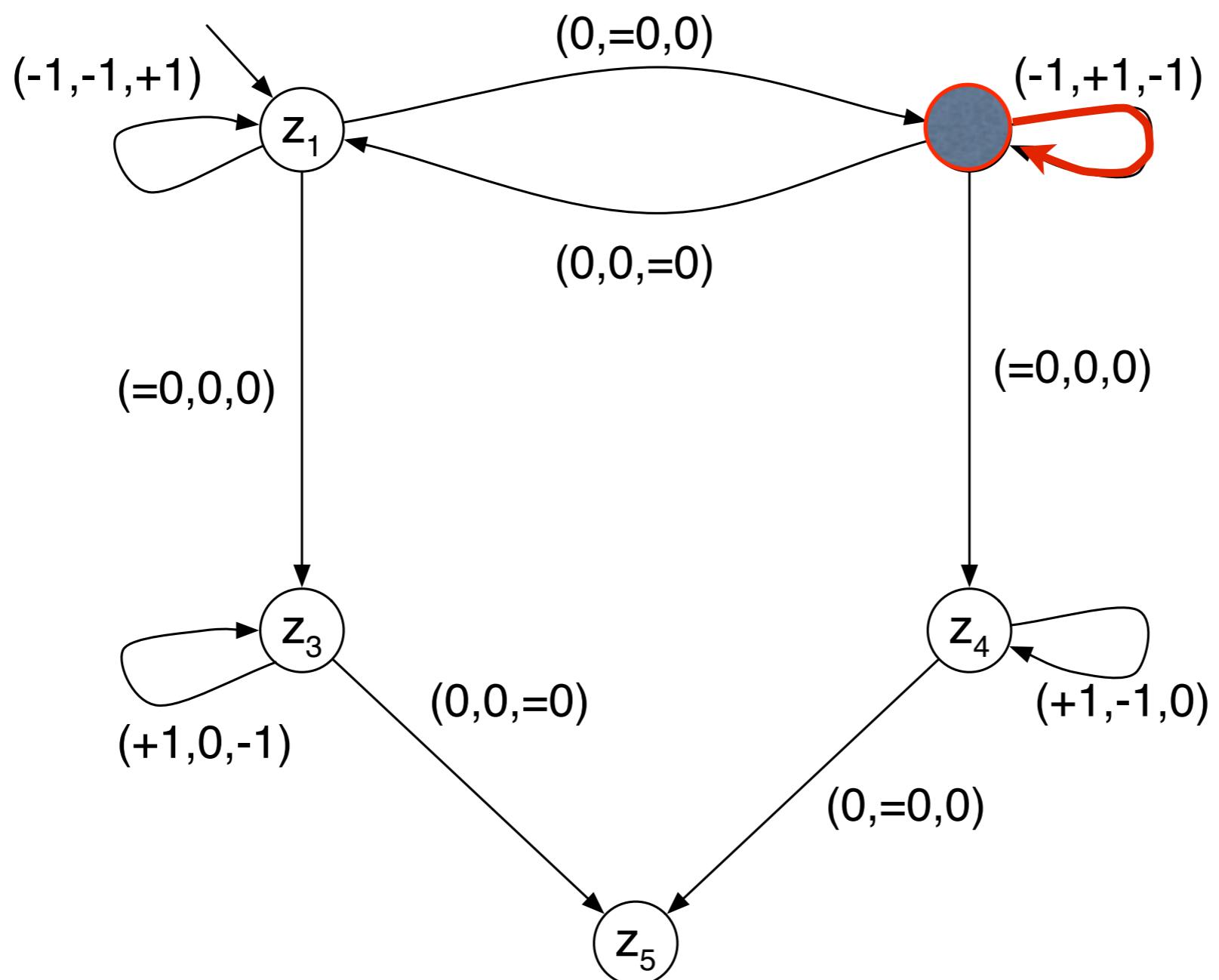
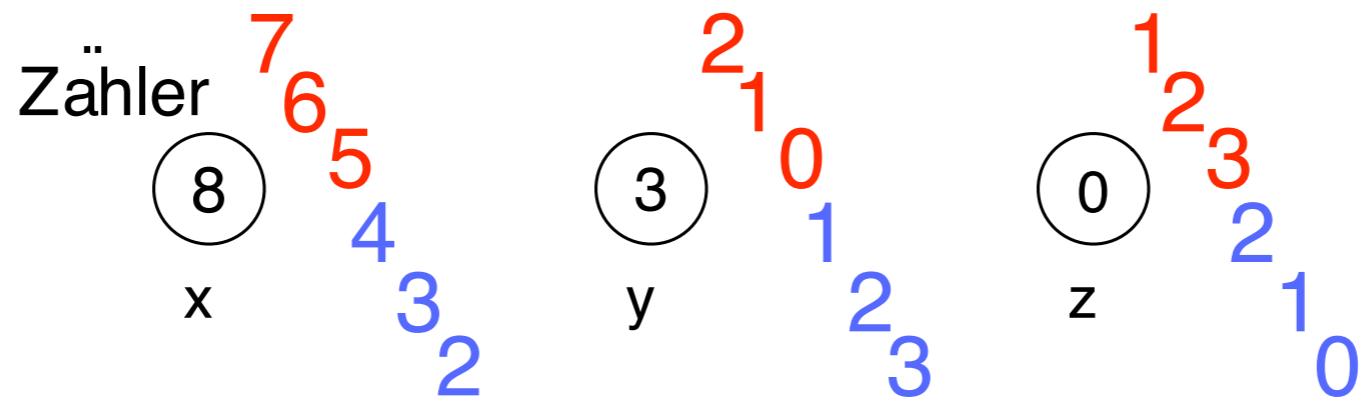
0

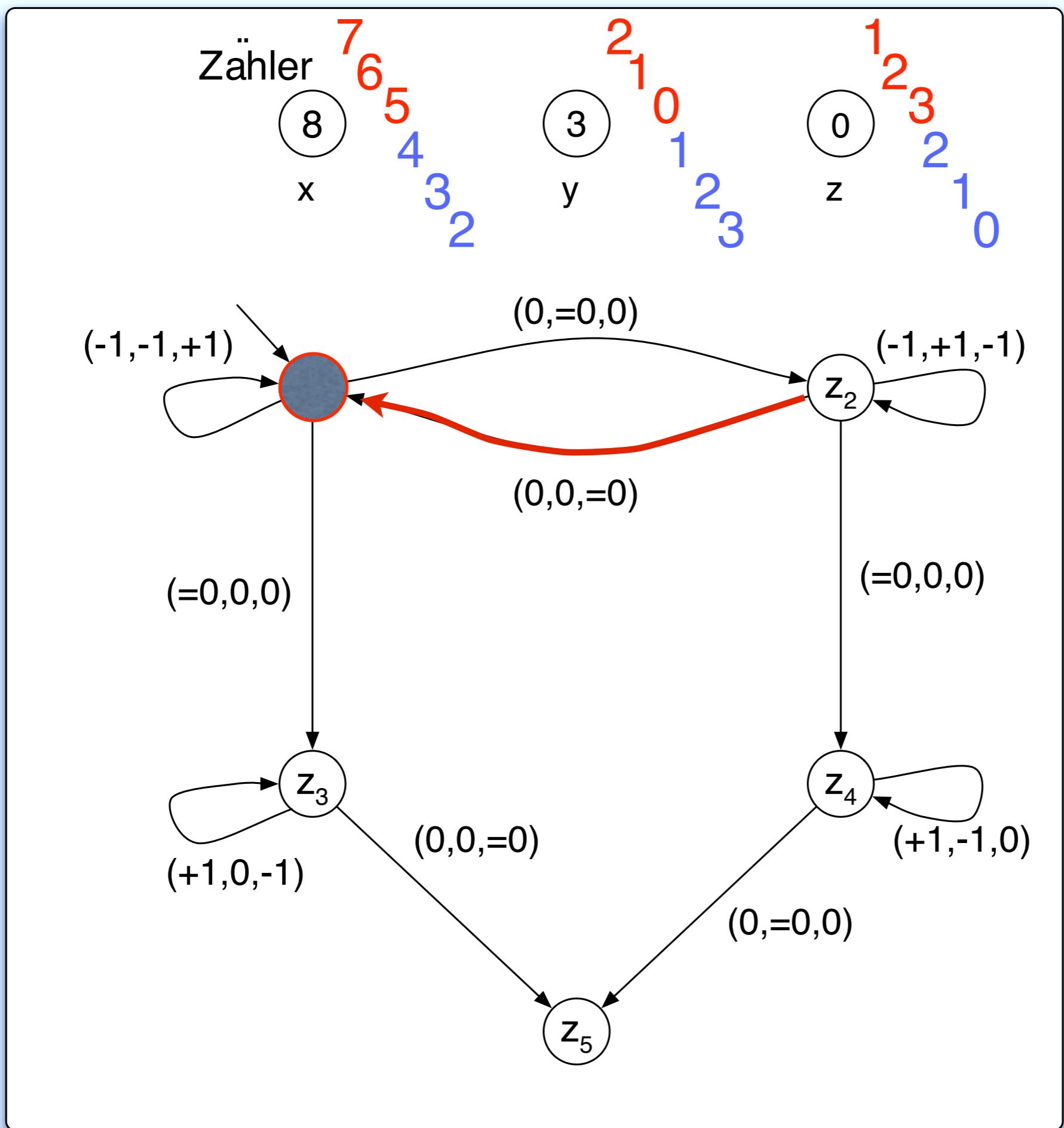
z

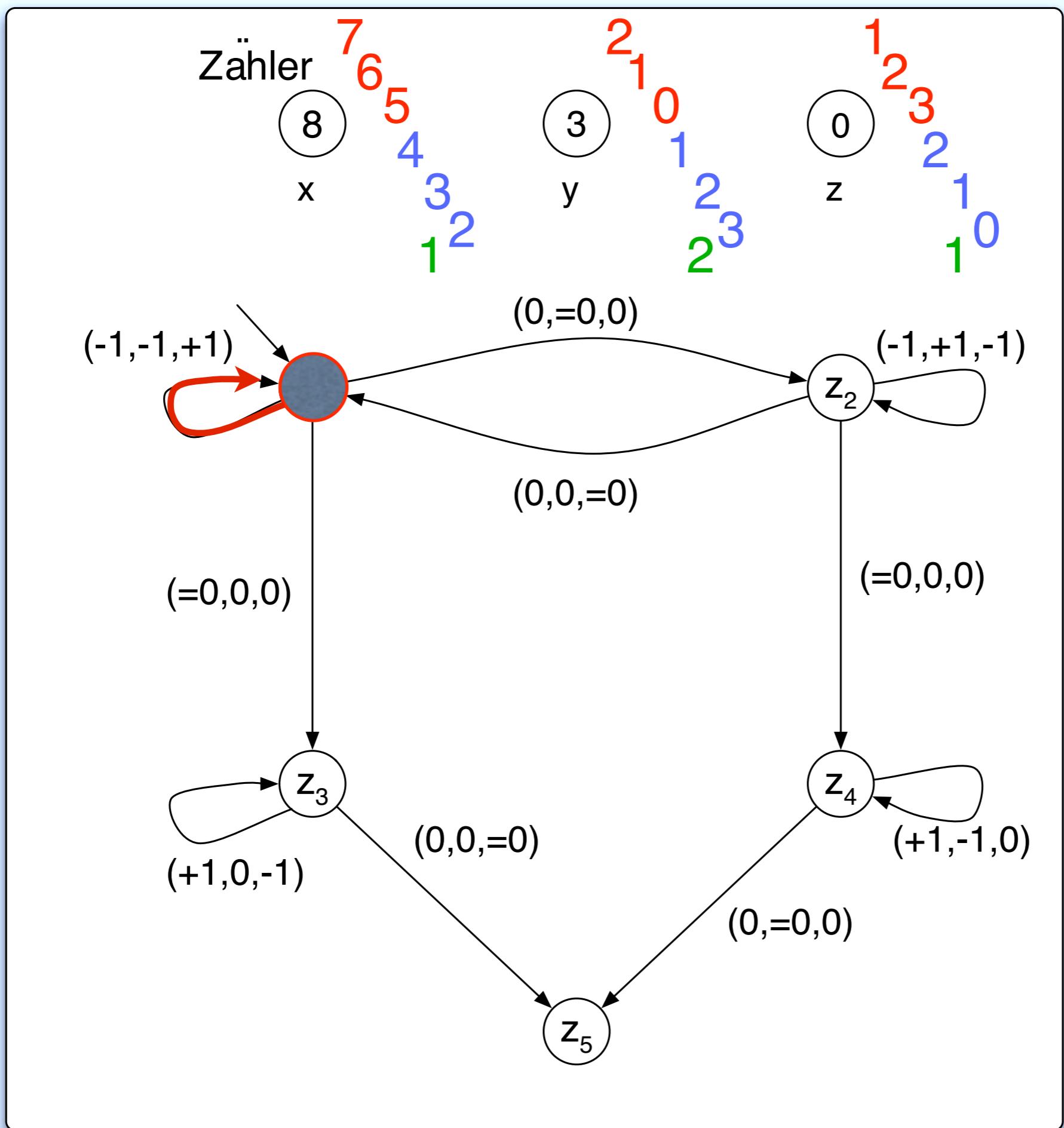


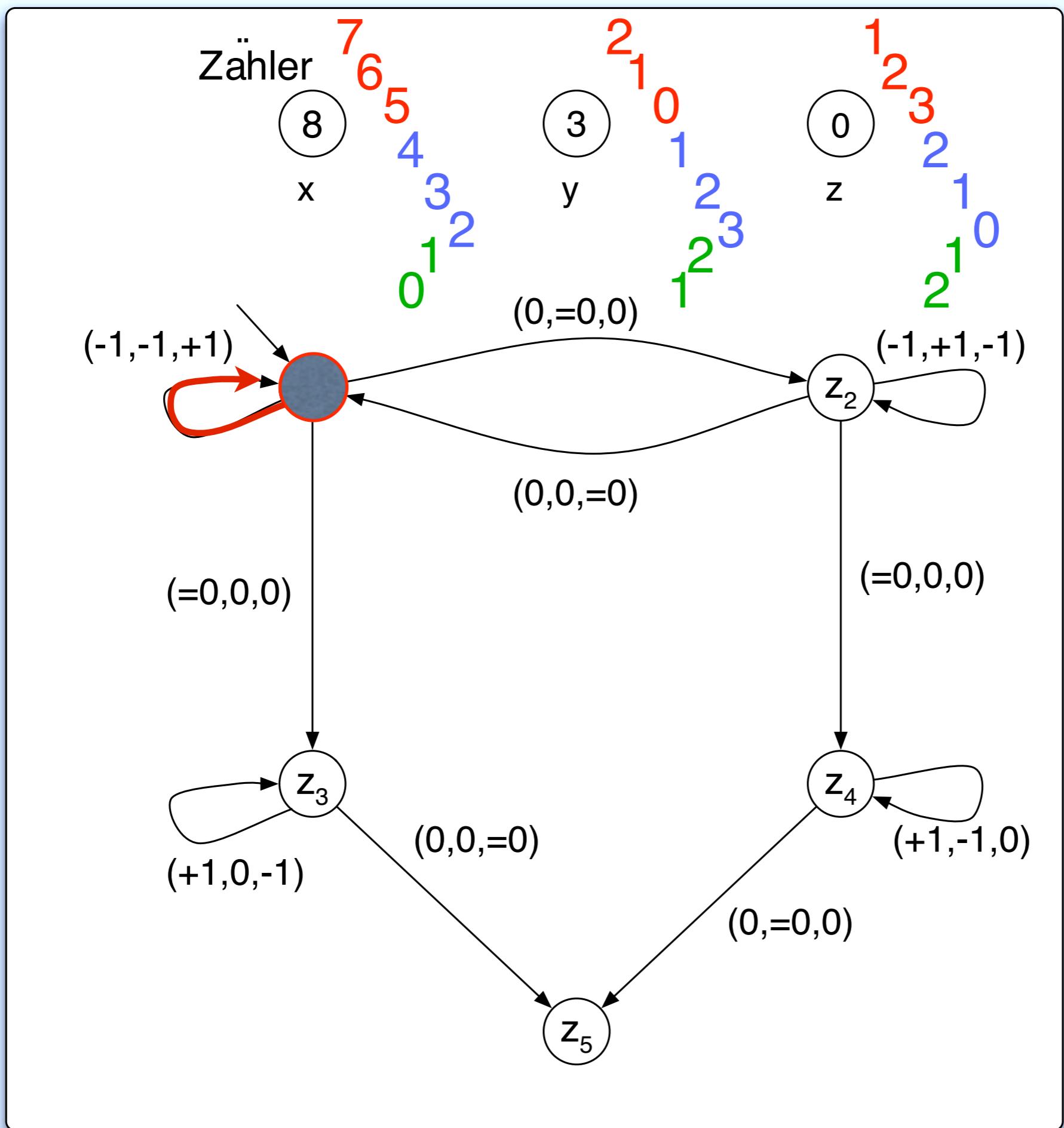


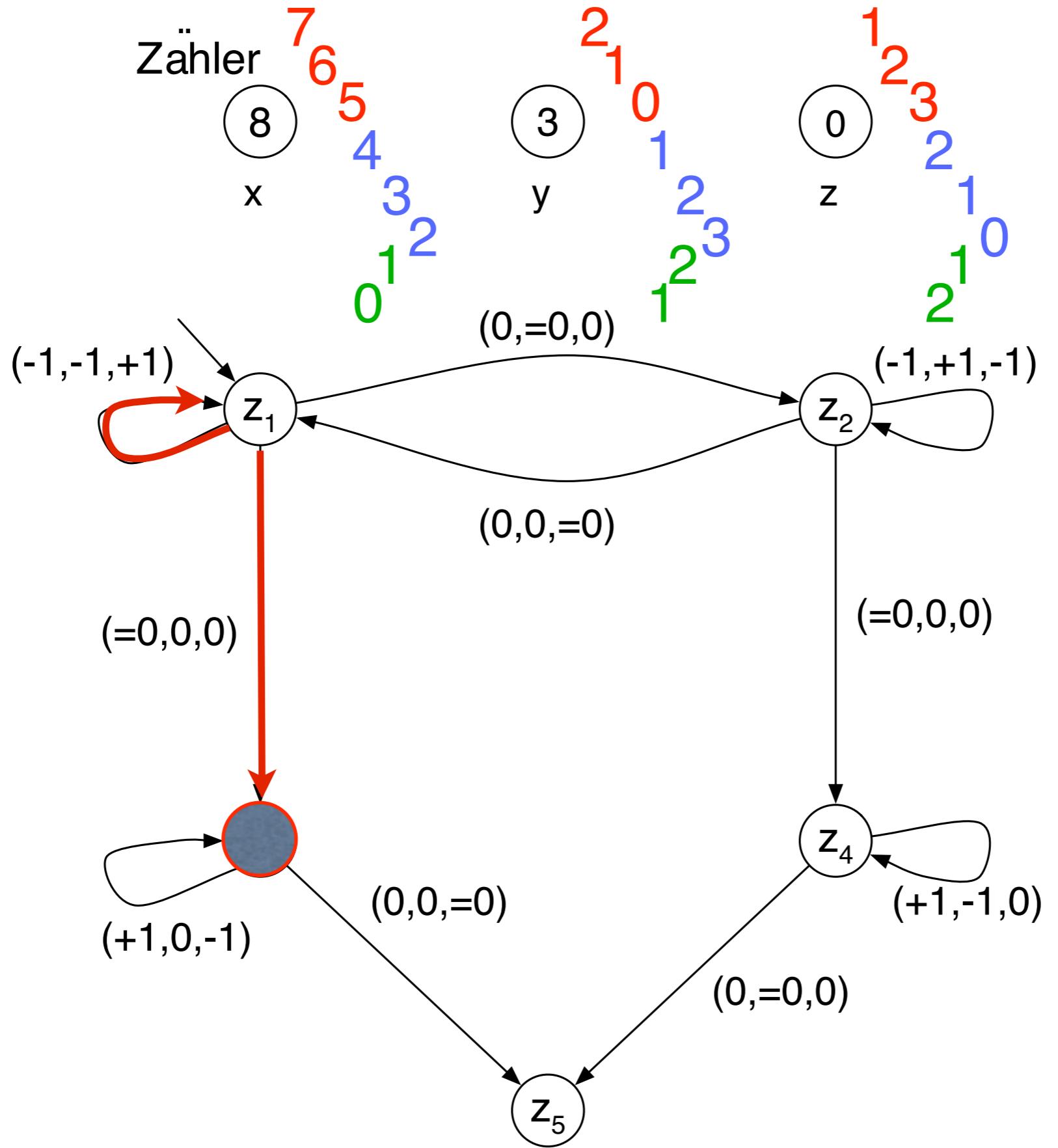


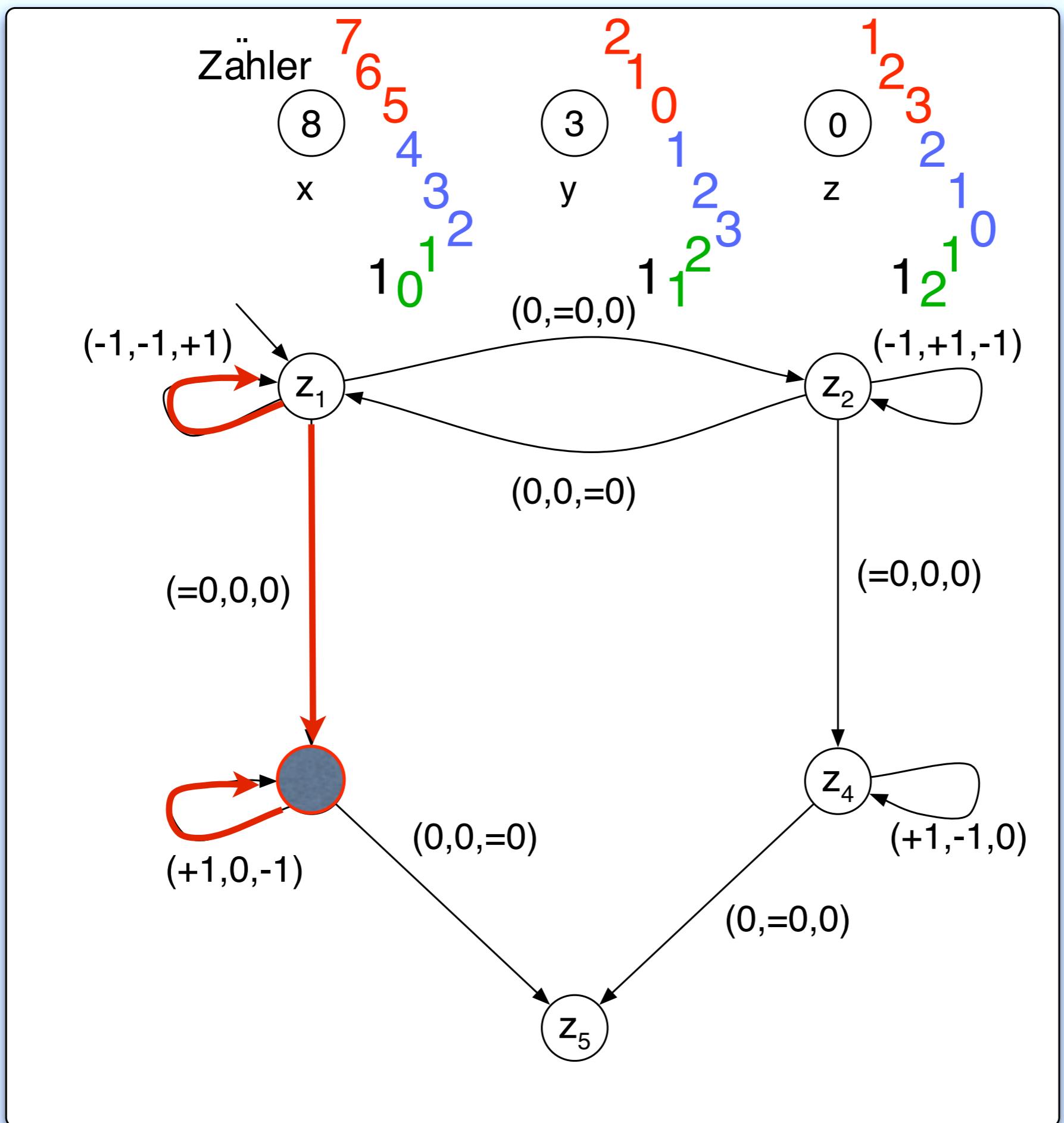


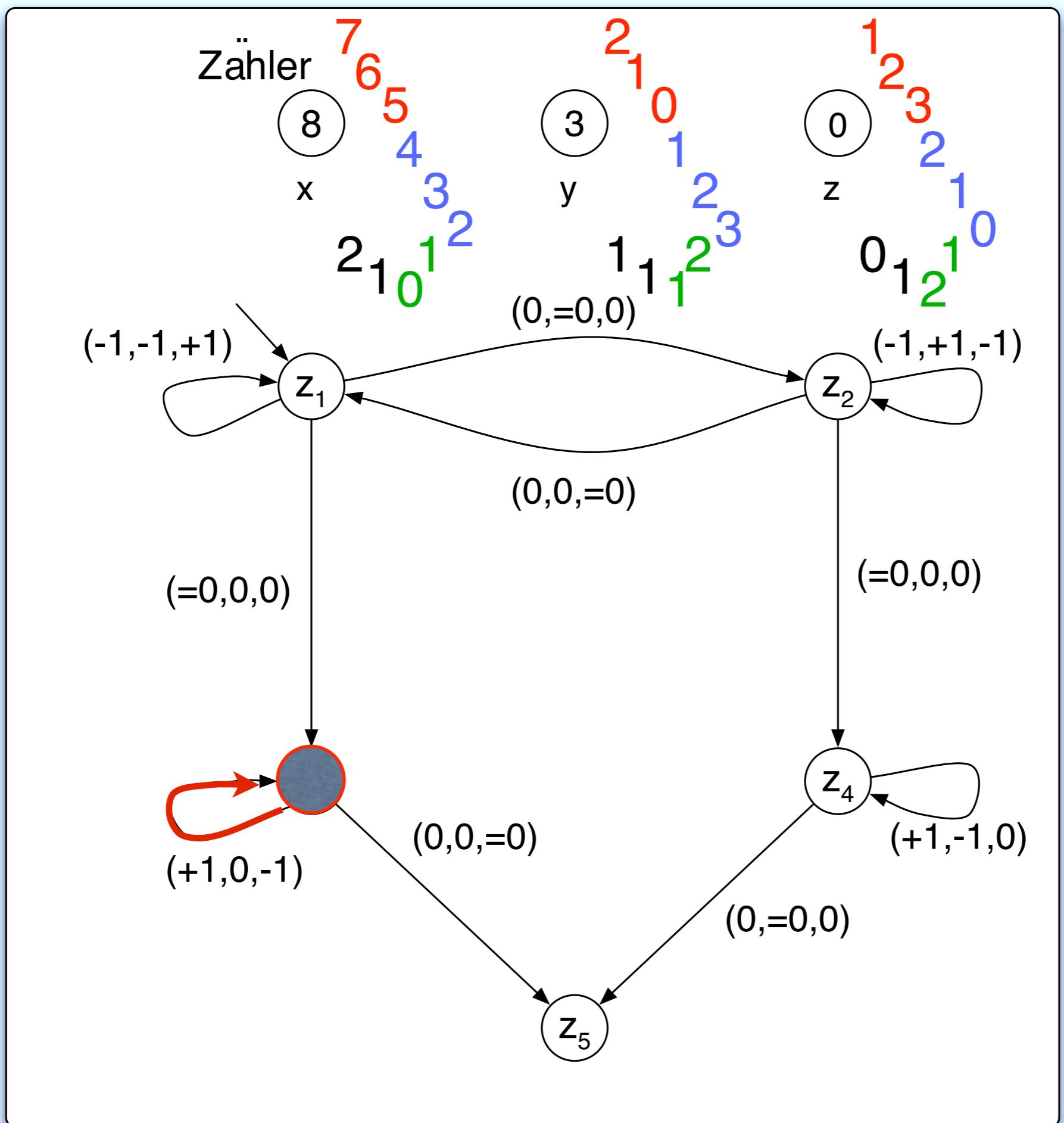


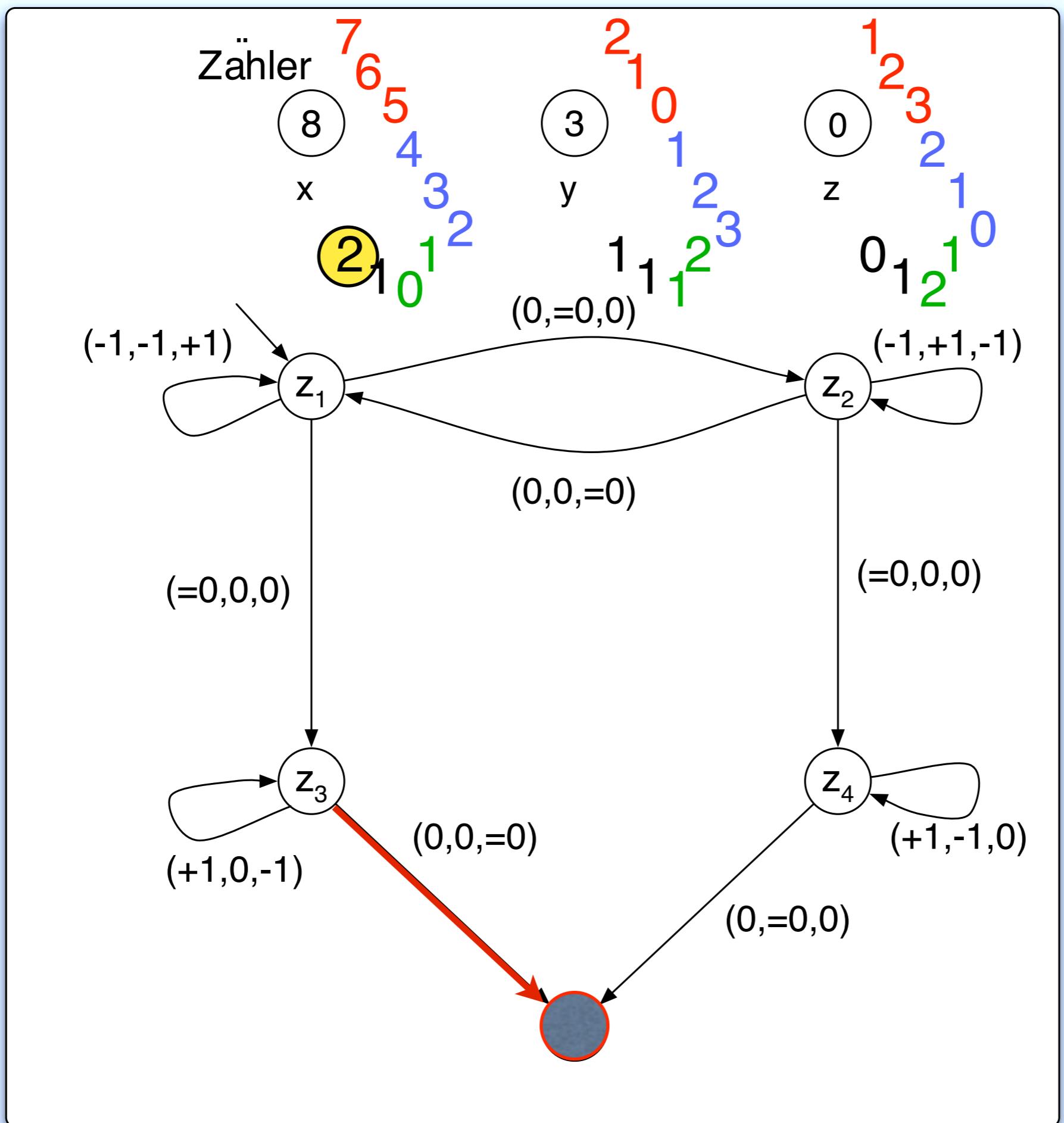












Zählerautomaten und P/T-Netze

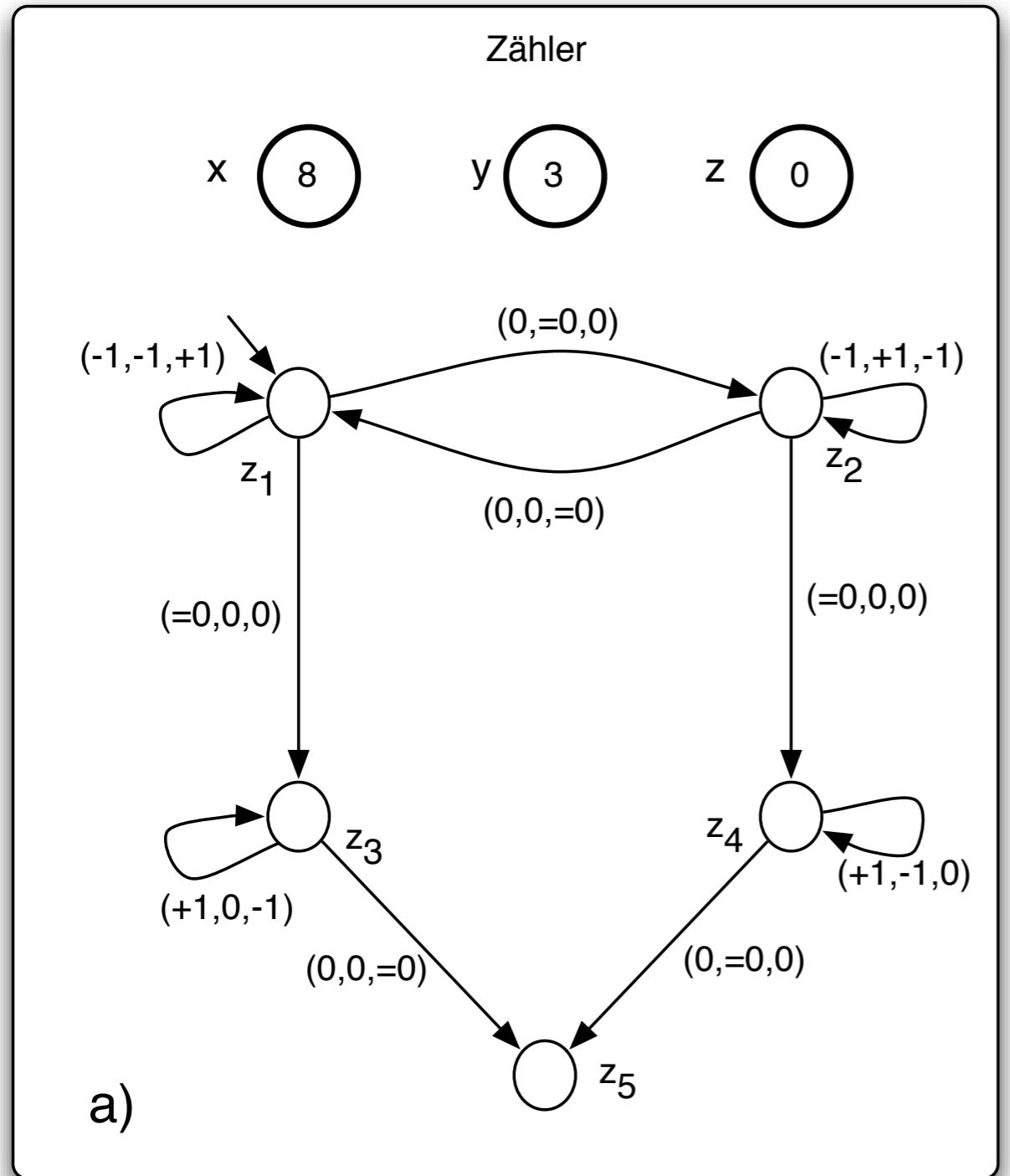
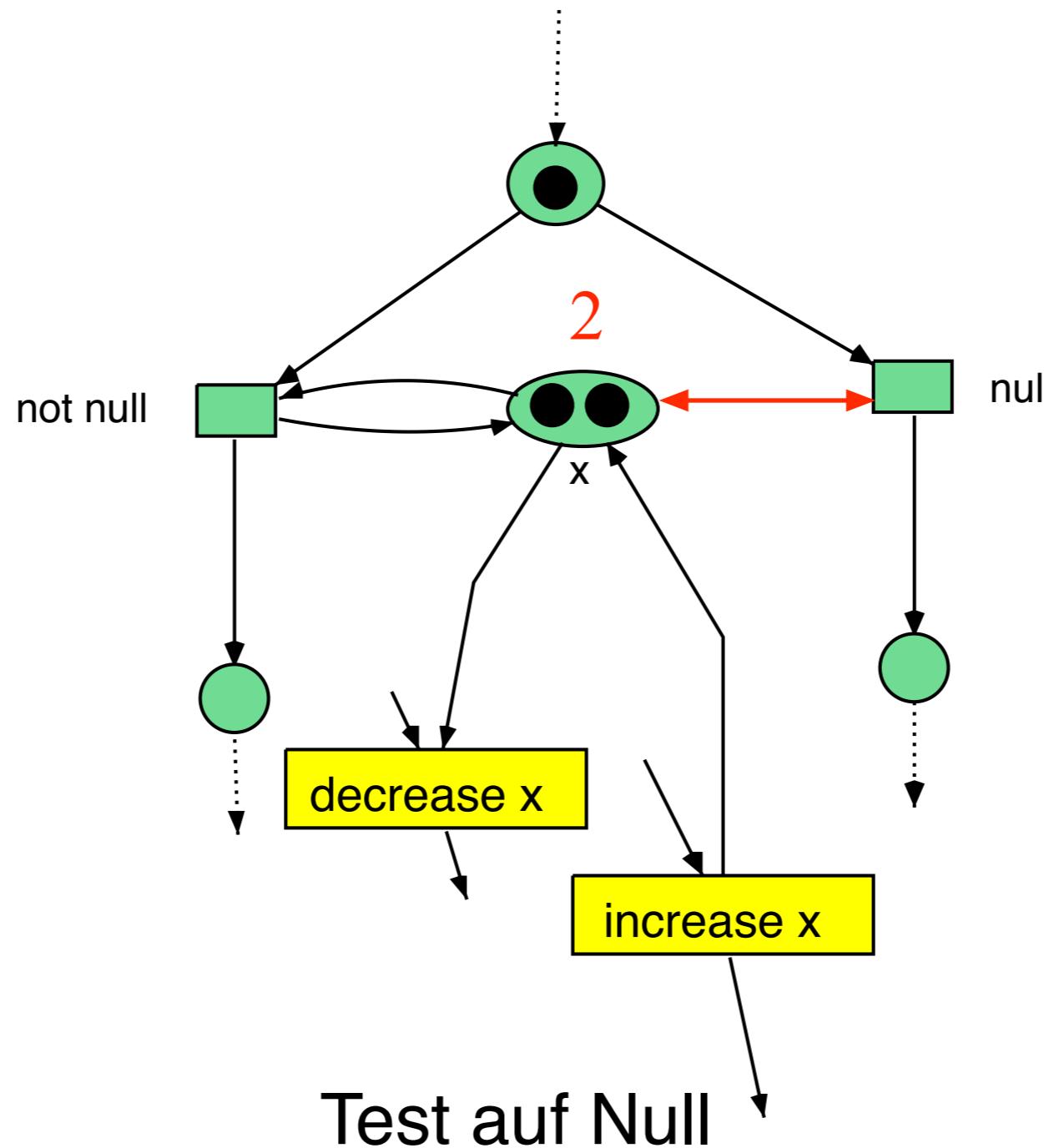
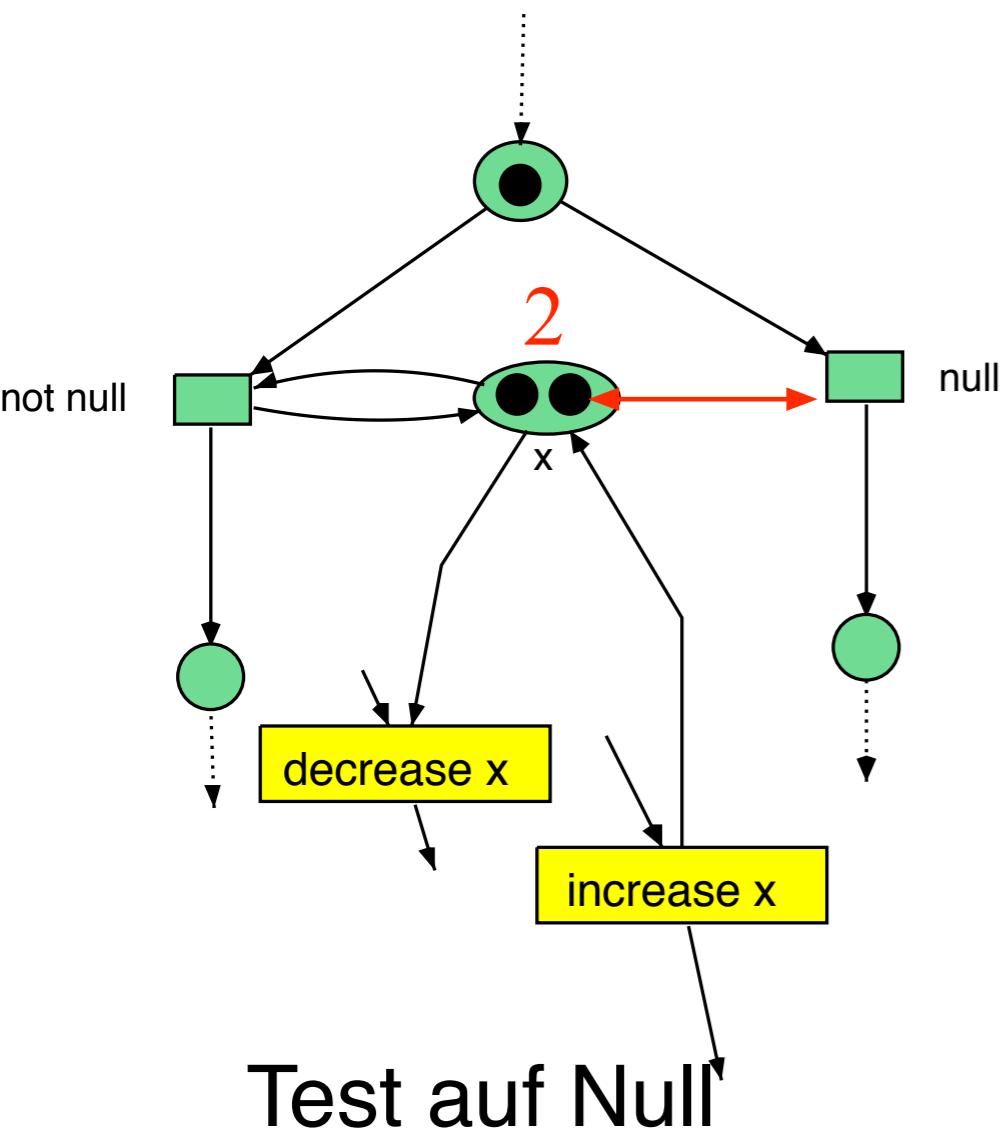


Abbildung 3.18: 3-Zählerautomat und P/T-Netz für $x := x \bmod y$

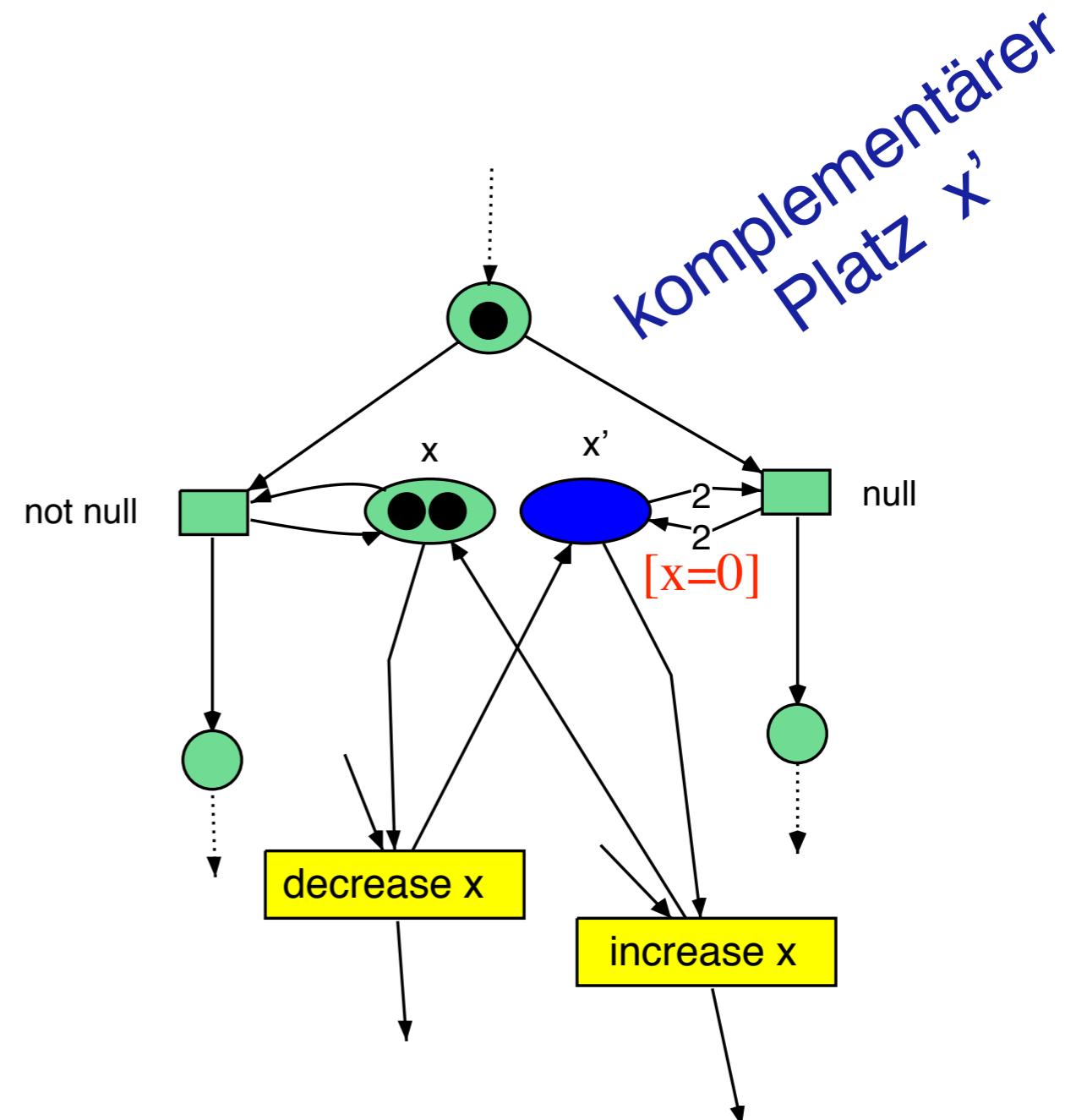
Sind Petrinetze Turing-mächtig ?



Sind Petrinetze Turing-mächtig ?



Test auf Null



Platz-Invariante: $\mathbf{m}(x) + \mathbf{m}(x') = k = 2$

Inhibitorkanten

Sind Petrinetze Turing-mächtig?

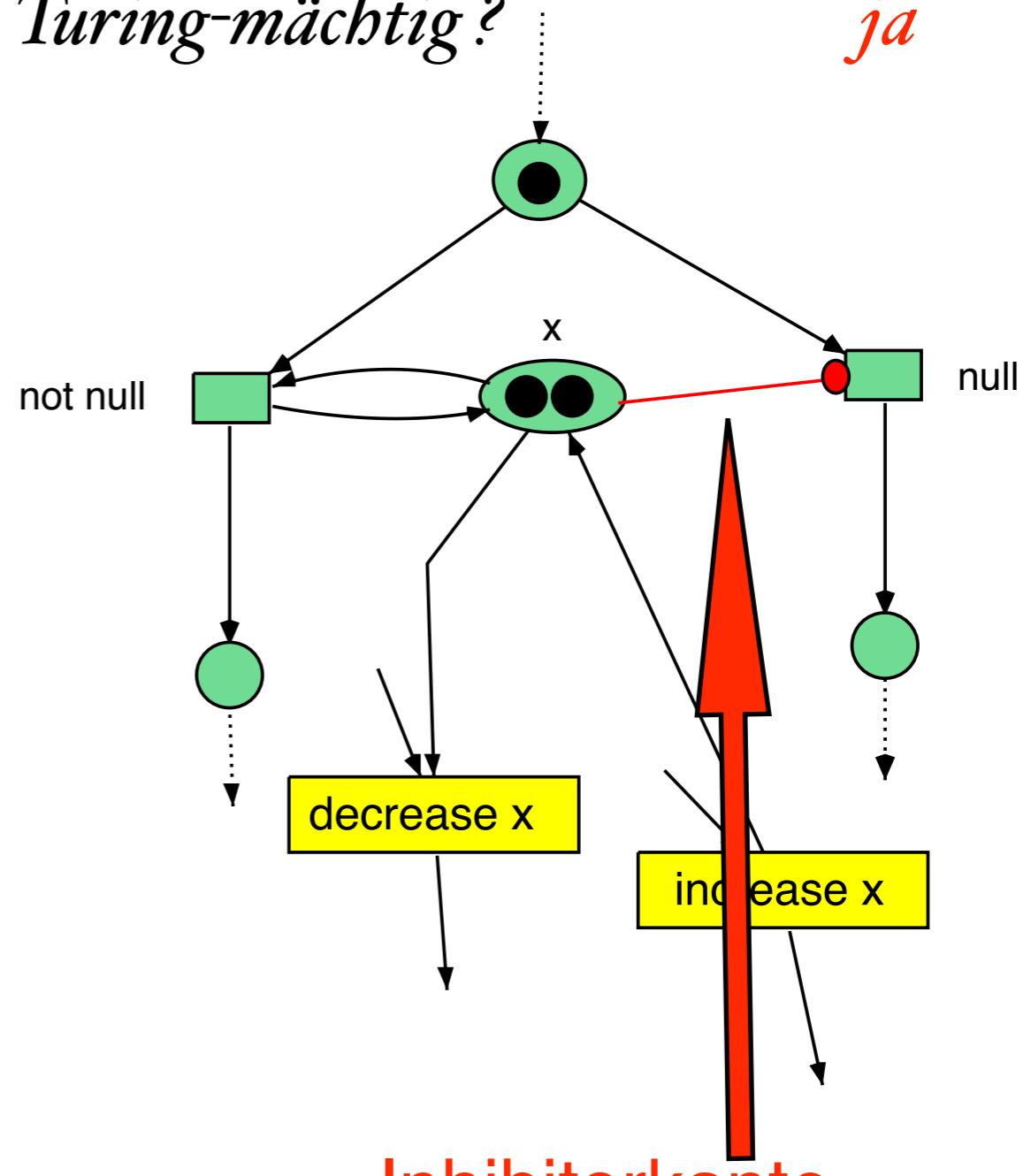
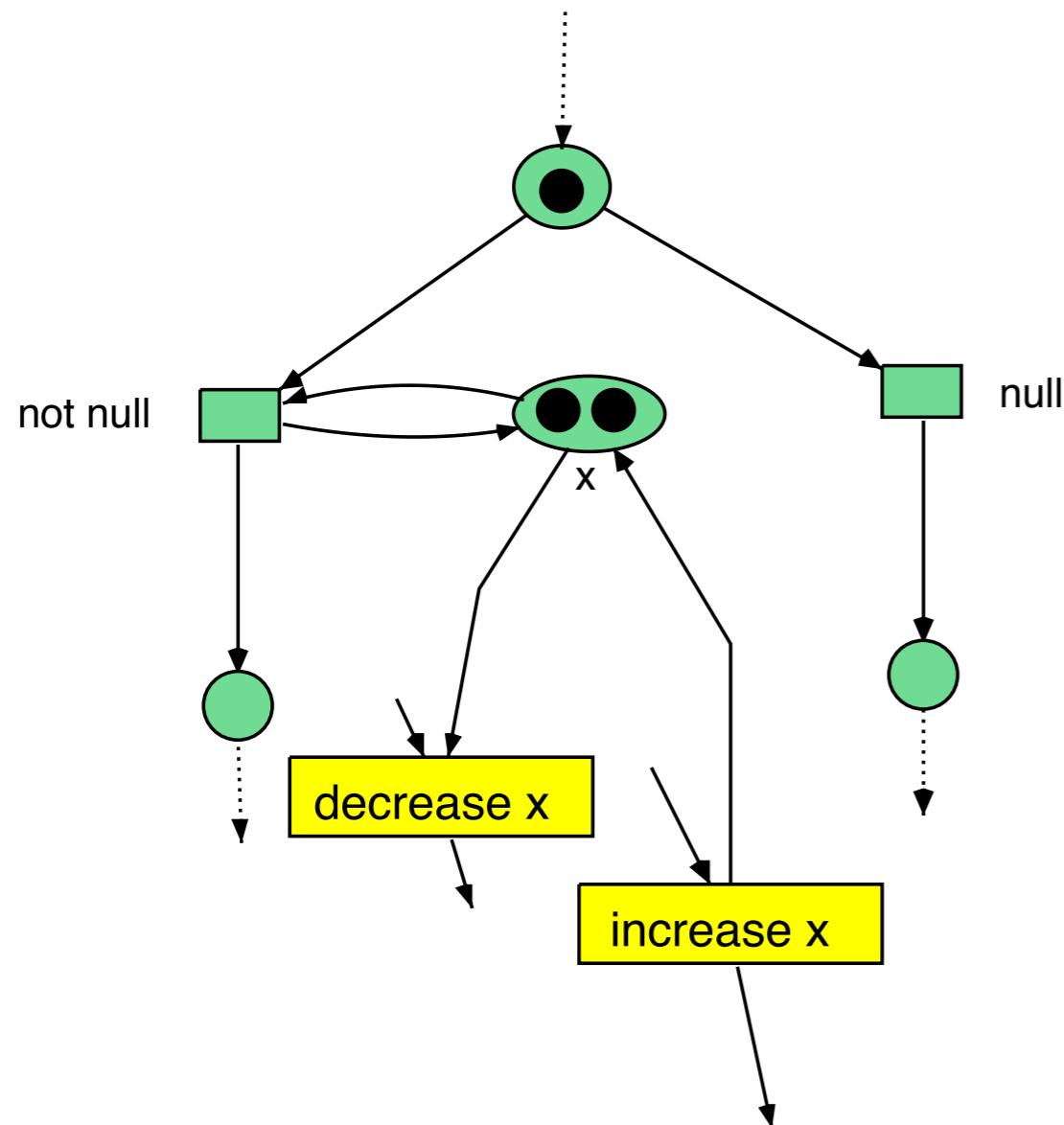
ja

Sind P/T-Netze Turing-mächtig?

nein

Sind P/T-Netze mit Inhibitorkanten Turing-mächtig?

ja



Übersicht

	Turing-mächtig?
P/T-Netze	nein
Inhibitor-Netze	ja
gefärbte Netze mit endlichen Farbmengen	nein
gefärbte Netze mit beliebigen Farbmengen	ja

Tabelle 3.2: Turing-Mächtigkeit von Petrinetzen

FGI 2

Daniel Moldt

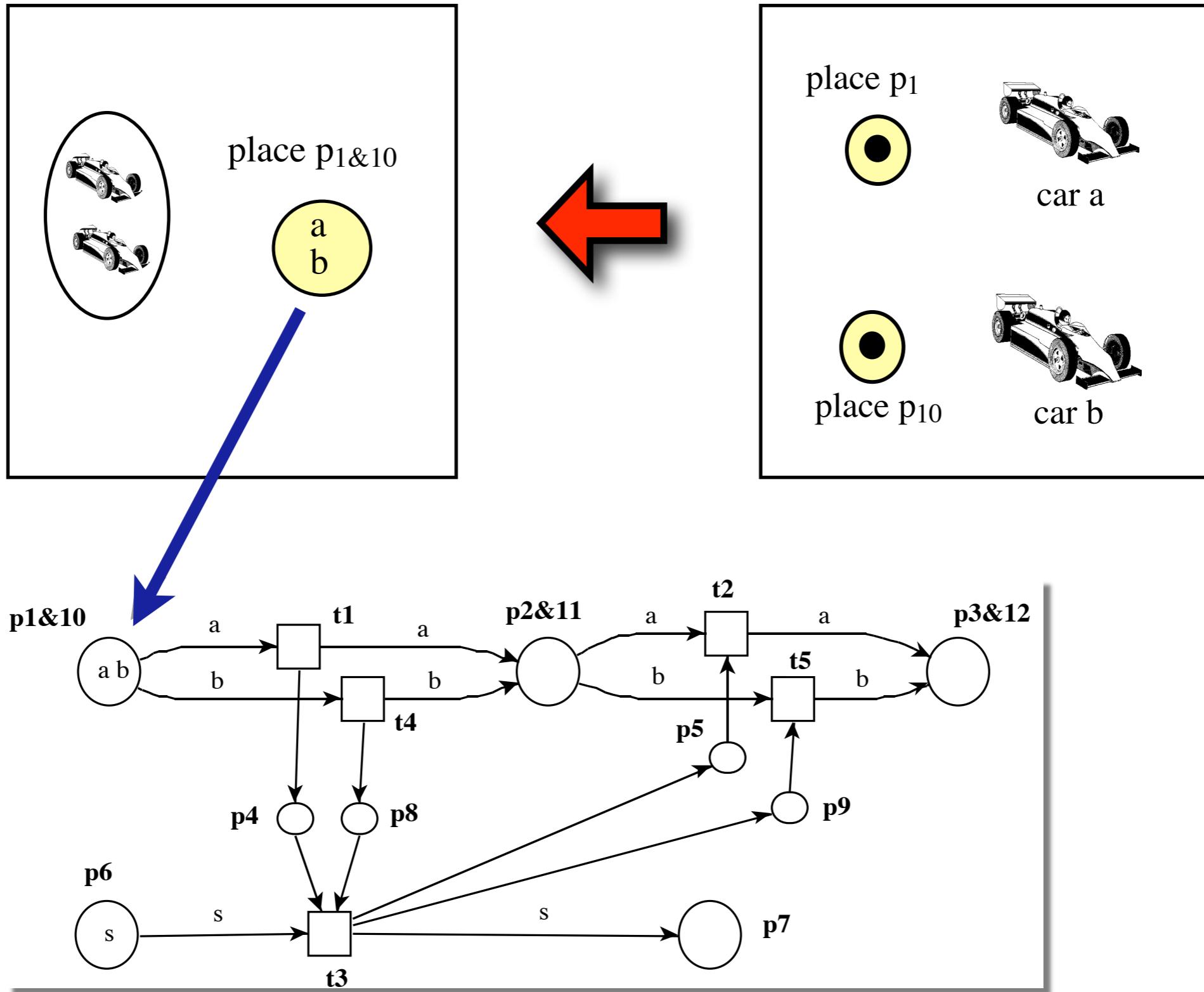
-> Höhere Petrinetze

FGI 2

Daniel Moldt

KKN

kantenkonstante Netze



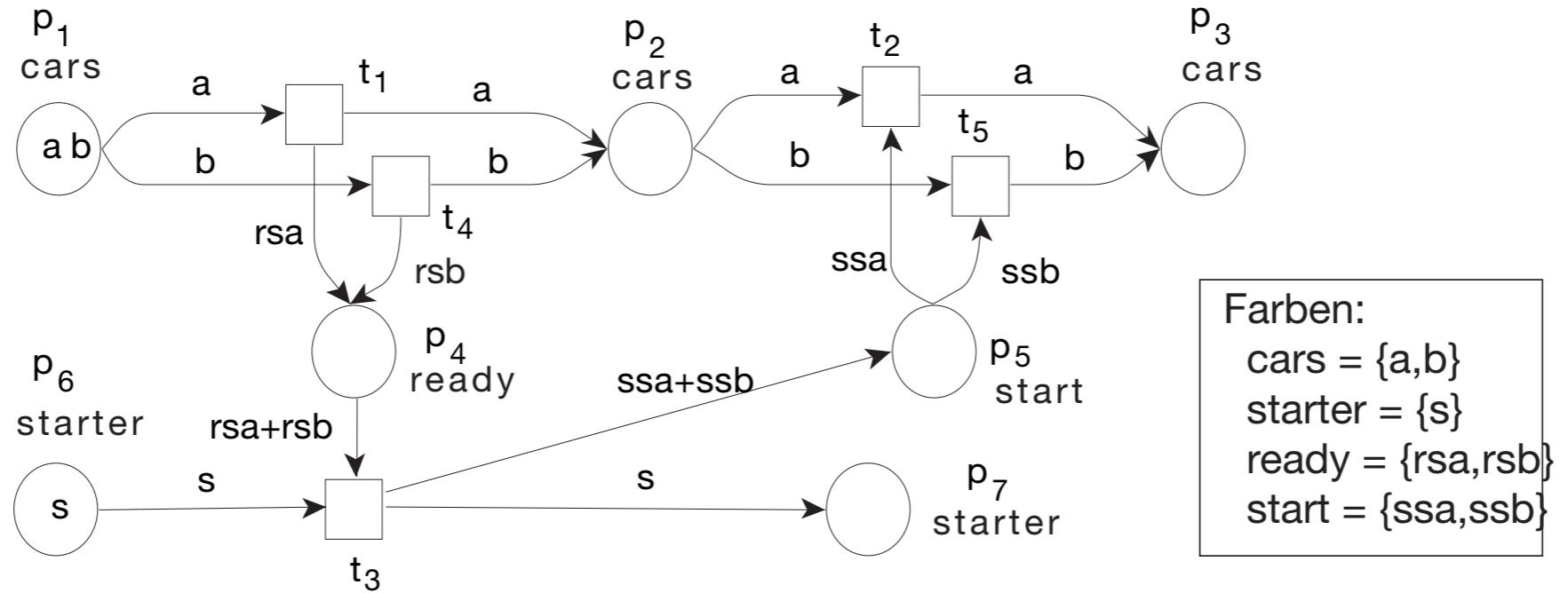
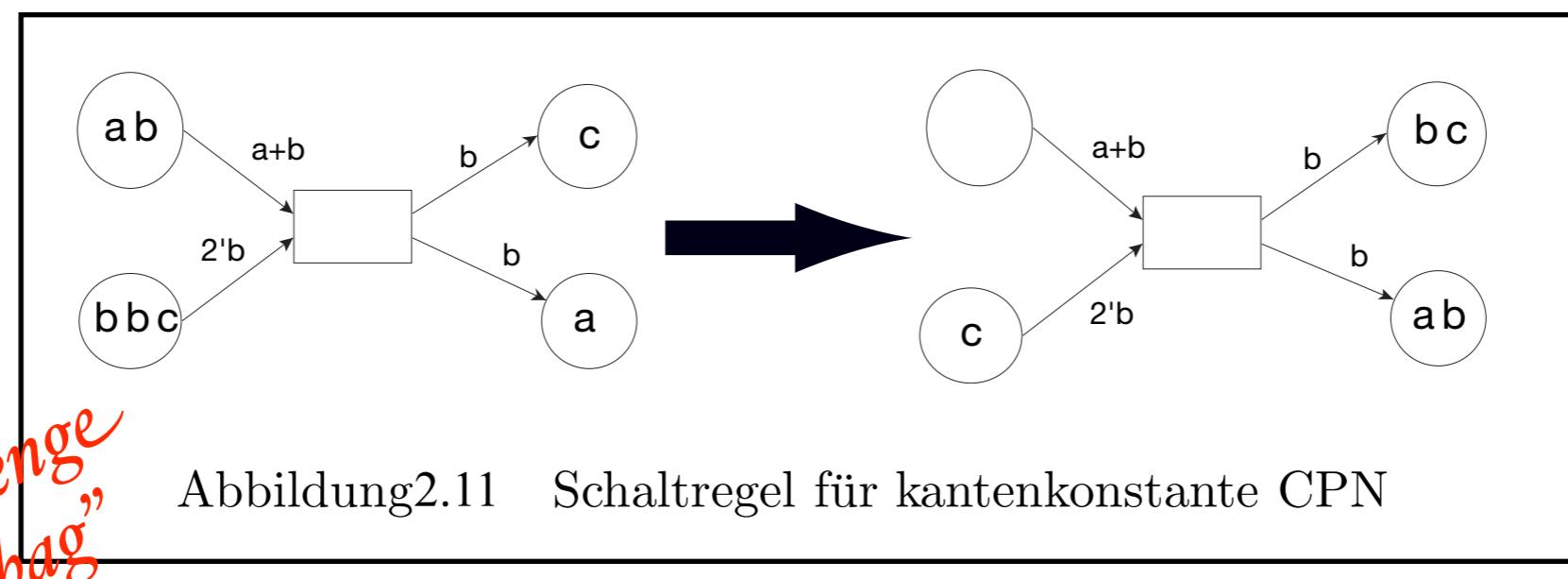
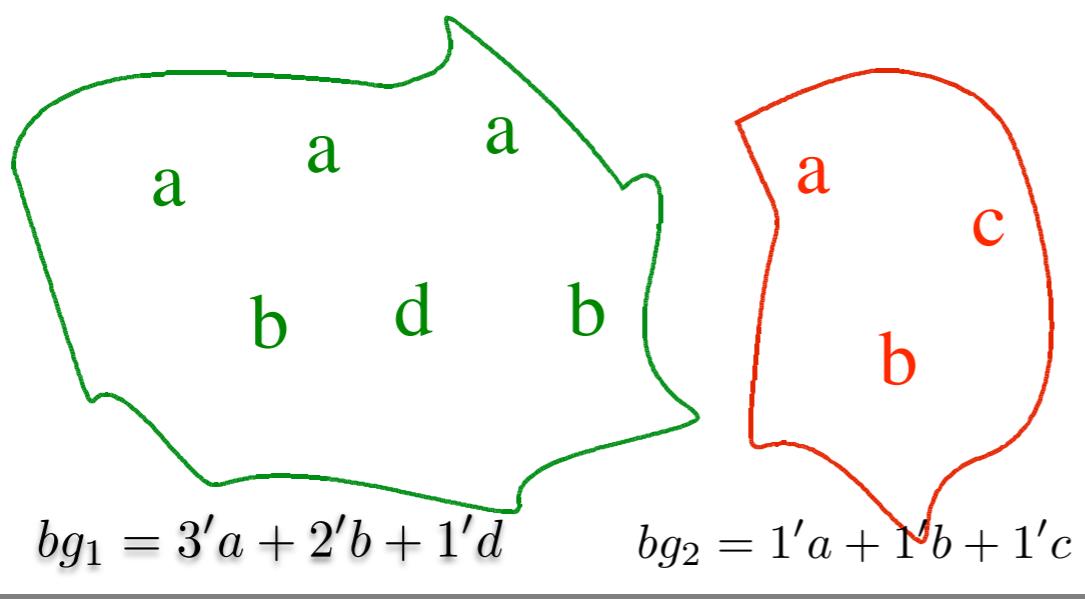


Abbildung 2.10: Das kantenkonstante gefärbte Netz \mathcal{N}_2



Multimengen, (multi-sets, bags)



$$\{a,a,a,b,b,d\}_b$$

$$3'a + 2'b + 0'c + 1'd$$

$$3'a + 2'b + d$$

$$\begin{aligned} &\neq \{a,a,a,b,b,d\} \\ &= \{a,b,d\} \end{aligned}$$

$$bg : A \rightarrow \mathbb{N}$$

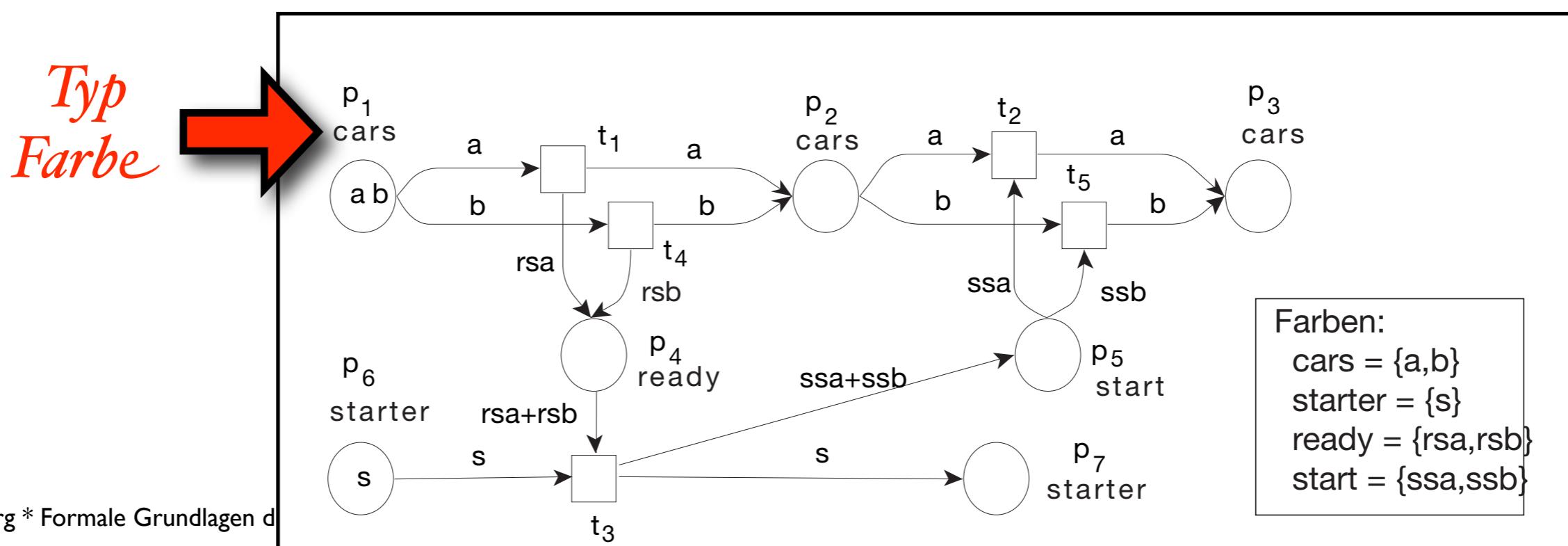
$$\sum_{a \in A} bg(a)'a$$

- $bg_1 + bg_2 := \sum_{a \in A} (bg_1(a) + bg_2(a))'a$ $\cong \text{Mengen-Vereinigung}$
 $4'a + 3'b + 1'c + 1'd$
- $bg_1 \leq bg_2 : \Leftrightarrow \forall a \in A : bg_1(a) \leq bg_2(a)$ $\cong \text{Mengen-Inklusion}$
- $bg_1 - bg_2 := \sum_{a \in A} (\text{Max}(bg_1(a) - bg_2(a), 0))'a$ und $2'a + 1'b + 1'd$ $\cong \text{Mengen-Differenz}$
- $|bg| := \sum_{a \in A} bg(a)$ ist die Mächtigkeit oder Kardinalität von bg
(nur definiert, falls die Summe endlich ist) und \emptyset bezeichnet die leere Multimenge (mit $|bg| = 0$). $\cong \text{Mengen-Kardinalität}$

$$|bg_1| = 6$$

Definition 8.3 Ein kantenkonstantes gefärbtes Petrinetz (KKN) wird als Tupel $\mathcal{N} = \langle P, T, F, \mathcal{C}, cd, W, \mathbf{m}_0 \rangle$ definiert, wobei

- (P, T, F) ein endliches Netz (Def. 3.1) ist,
- \mathcal{C} ist eine Menge von Farbenmengen,
- $cd: P \rightarrow \mathcal{C}$ ist die Farbzueisungsabbildung (colour domain mapping). Sie wird durch $cd: F \rightarrow \mathcal{C}$, $cd(x, y) := \text{if } x \in P \text{ then } cd(x) \text{ else } cd(y)$ für auf F erweitert.
- $W : F \rightarrow \text{Bag}(\bigcup \mathcal{C})$ mit $W(x, y) \in \text{Bag}(cd(x, y))$ heißt Kantengewichtung.
- $\mathbf{m}_0 : P \rightarrow \text{Bag}(\bigcup \mathcal{C})$ mit $\mathbf{m}_0(p) \in \text{Bag}(cd(p))$ für alle $p \in P$ ist die Anfangsmarkierung. Darstellung als Abbildung oder Vektor möglich

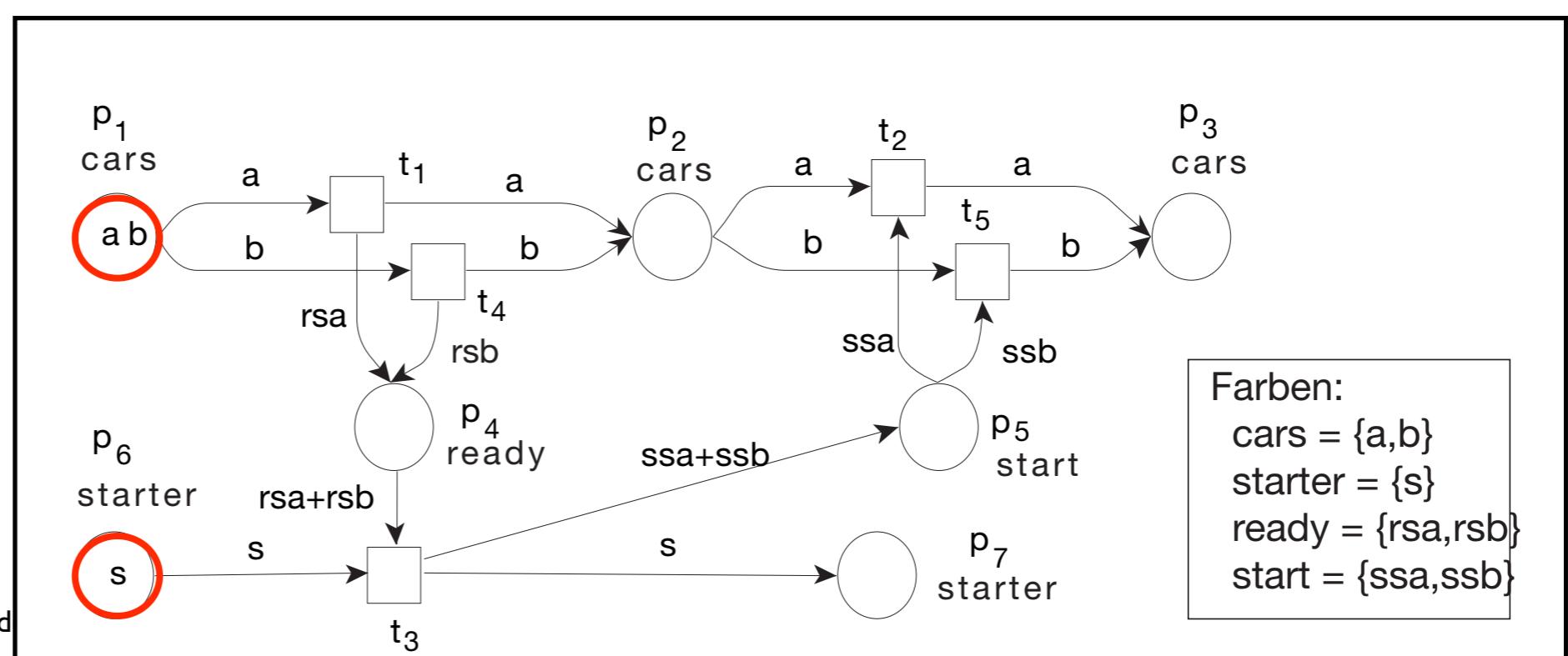
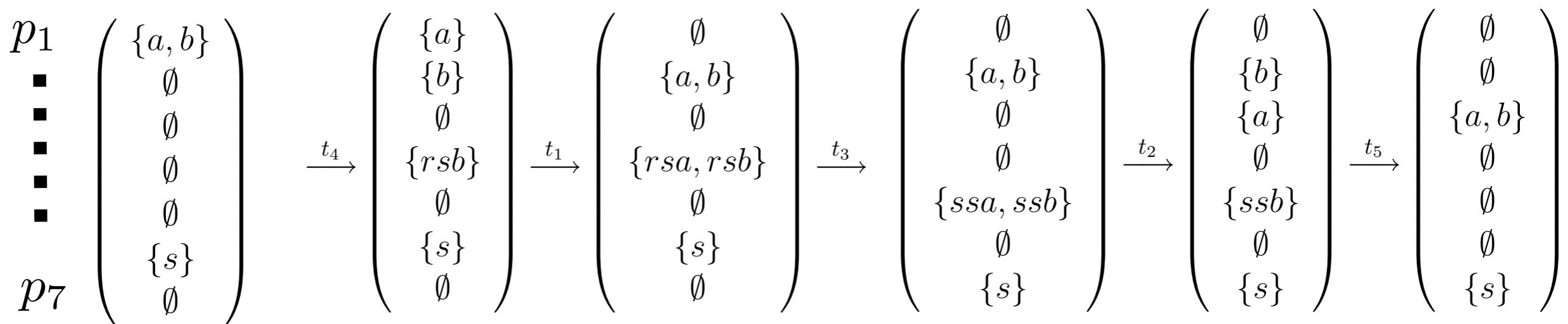


Darstellung als Abbildung:

$$\begin{aligned}\mathbf{m}_0(p_1) &= \{a, b\}_b \\ \mathbf{m}_0(p_6) &= \{s\}_b\end{aligned}$$

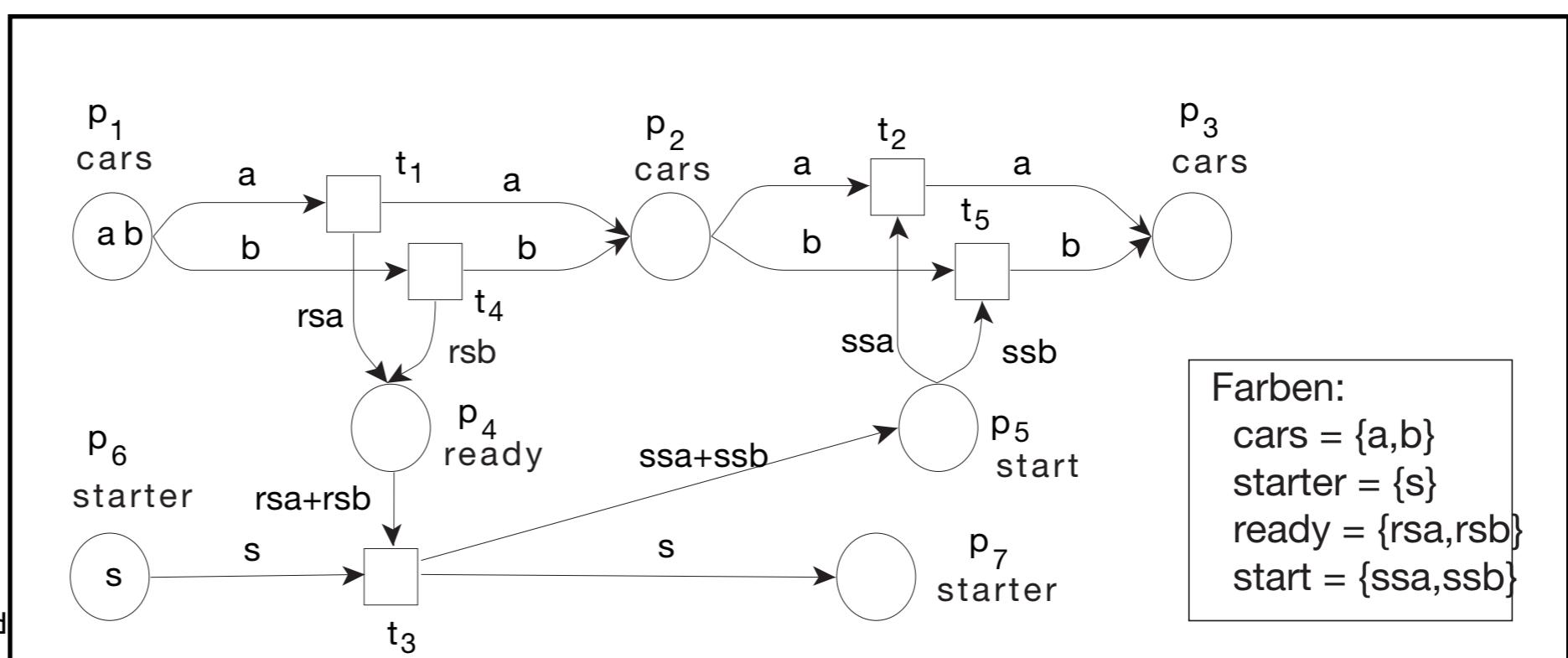
$$\mathbf{m}_0(p_i) = \emptyset$$

Darstellung als Vektor:



Definition 8.4 a) Die Markierung eines KKN \mathcal{N} = $\langle P, T, F, \mathcal{C}, cd, W, \mathbf{m}_0 \rangle$ ist ein Vektor \mathbf{m} mit $\mathbf{m}(p) \in Bag(cd(p))$ für jedes $p \in P$ (auch als Abbildung $\mathbf{m} : P \rightarrow Bag(\bigcup \mathcal{C})$ mit $\mathbf{m}(p) \in Bag(cd(p))$ für jedes $p \in P$ aufzufassen).

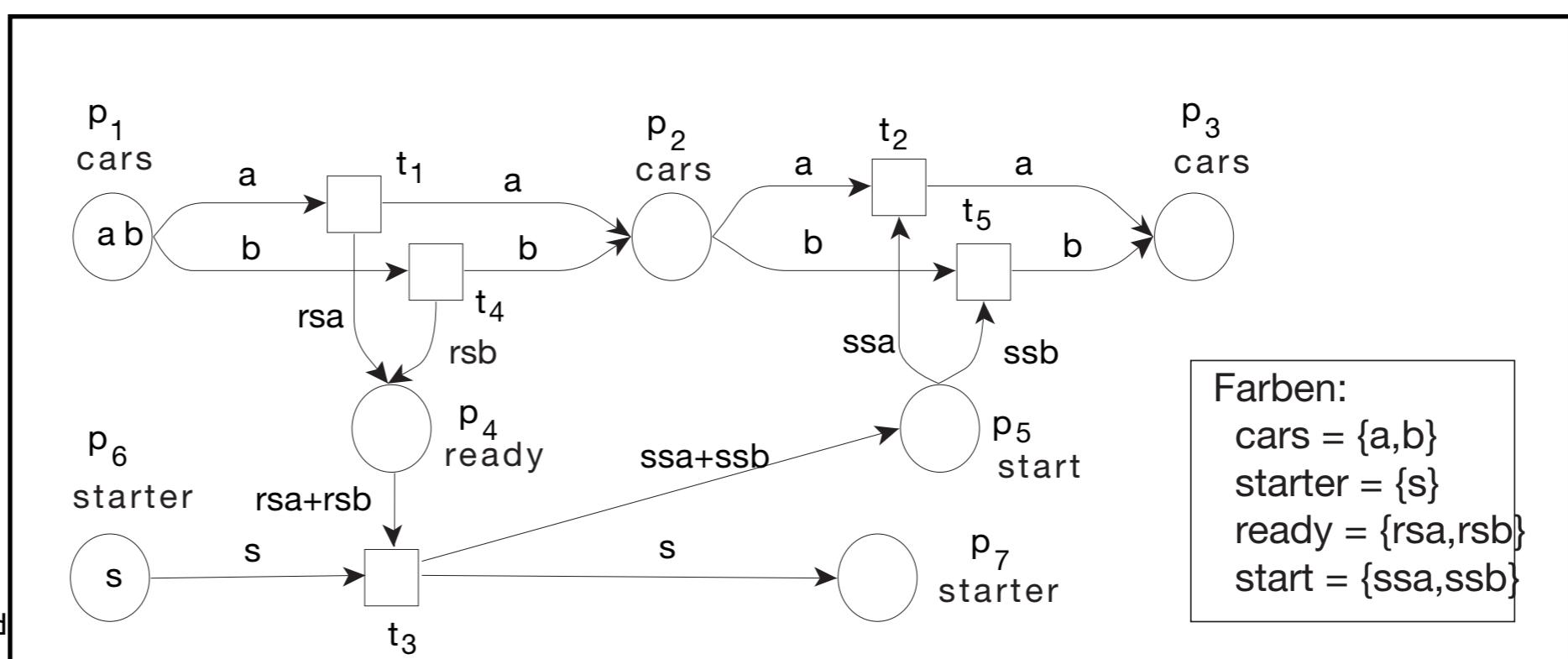
b) Eine Transition $t \in T$ heißt aktiviert in einer Markierung \mathbf{m} falls $\forall p \in \bullet t. \mathbf{m}(p) \geq W(p, t)$ (als Relation: $\mathbf{m} \xrightarrow{t}$).



c) Es sei $\widetilde{W}(p, t) := \begin{cases} W(p, t) & \text{falls } (p, t) \in F, \\ \emptyset & \text{sonst.} \end{cases}$

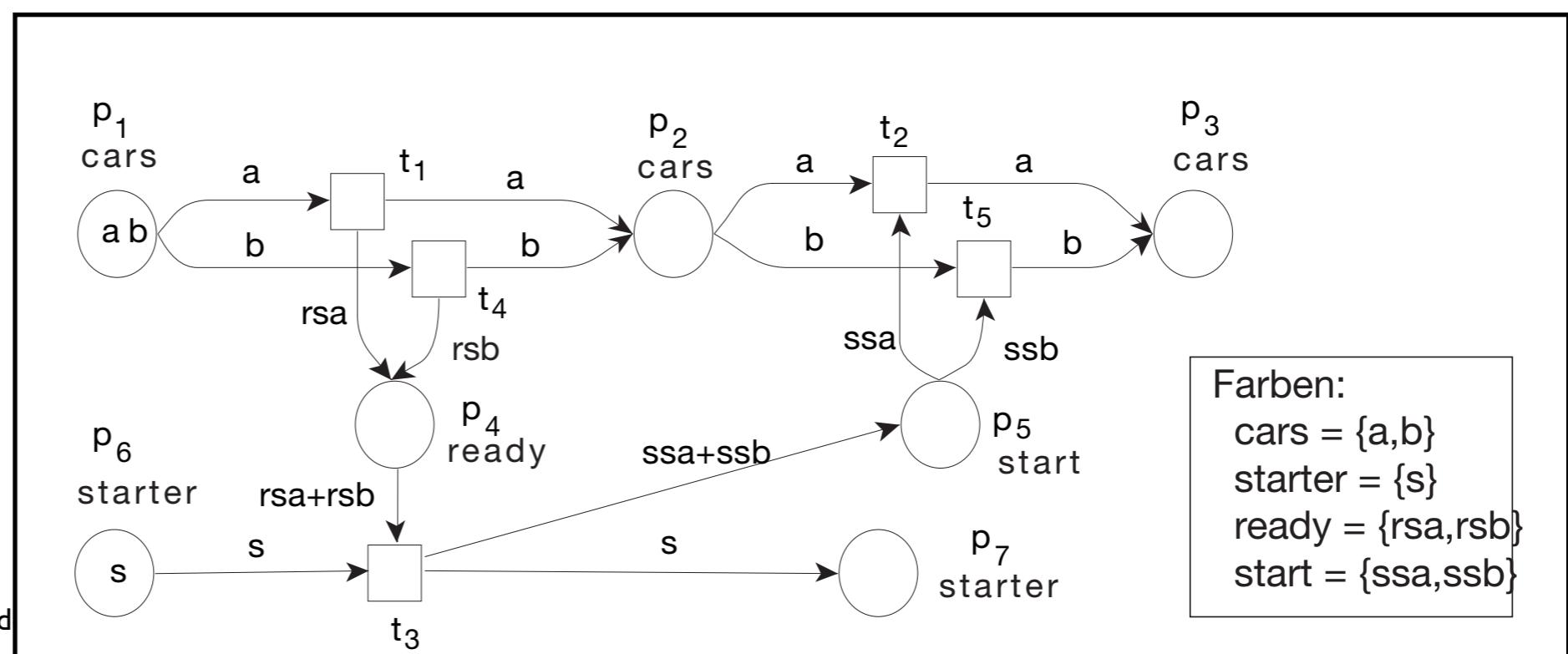
und entsprechend $\widetilde{W}(t, p) := \begin{cases} W(t, p) & \text{falls } (t, p) \in F, \\ \emptyset & \text{sonst.} \end{cases}$

Ist t in \mathbf{m} aktiviert, dann ist die Nachfolgemarkierung definiert durch $\mathbf{m} \xrightarrow{t} \mathbf{m}' \Leftrightarrow \forall p \in P. (\mathbf{m}(p) \geq \widetilde{W}(p, t) \wedge \mathbf{m}'(p) = \mathbf{m}(p) - \widetilde{W}(p, t) + \widetilde{W}(t, p))$. (Beachte, dass es sich um Multimengenoperationen handelt!).



d) Definiert man $W(\bullet, t) := (\widetilde{W}(p_1, t), \dots, \widetilde{W}(p_{|P|}, t))$ als Vektor der Länge $|P|$ (und sinngemäß ebenso $W(t, \bullet)$), dann kann die Nachfolgemarkierung kürzer durch Vektoren definiert werden:
 $\mathbf{m} \xrightarrow{t} \mathbf{m}' \Leftrightarrow \mathbf{m} \geq W(\bullet, t) \wedge \mathbf{m}' = \mathbf{m} - W(\bullet, t) + W(t, \bullet)$.

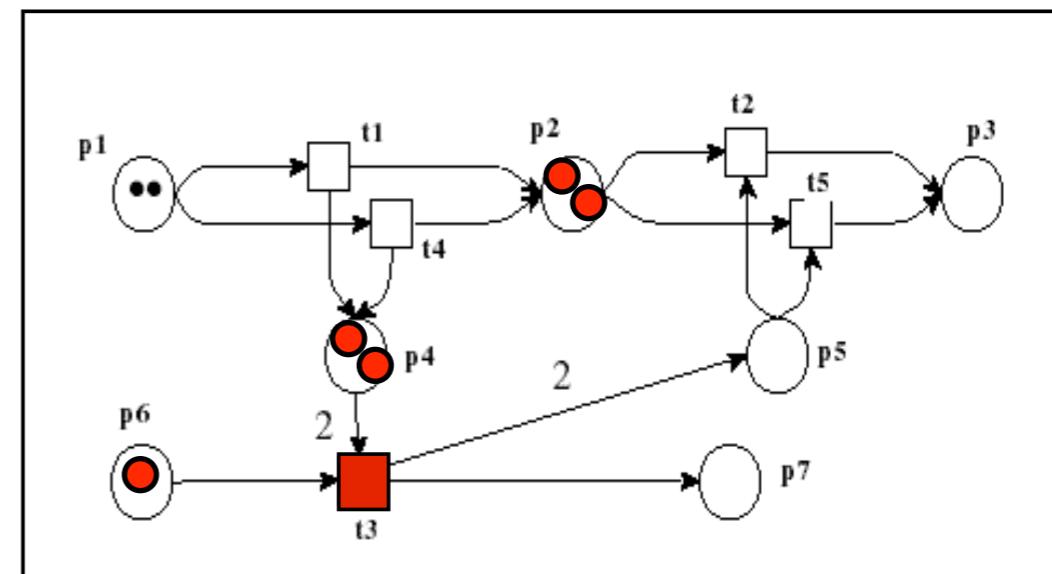
Dabei sind die Multimengenoperatoren komponentenweise auf Vektoren zu erweitern.



$W(\bullet, t_3) \leq m$

	t_1	t_2	t_3
p_1			
p_2			
p_3			
p_4		2	
p_5			
p_6			1
p_7			

\nwarrow



$$\begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{t_1} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 2 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{t_3} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 2 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{t_2} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{t_2} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

m

$+$

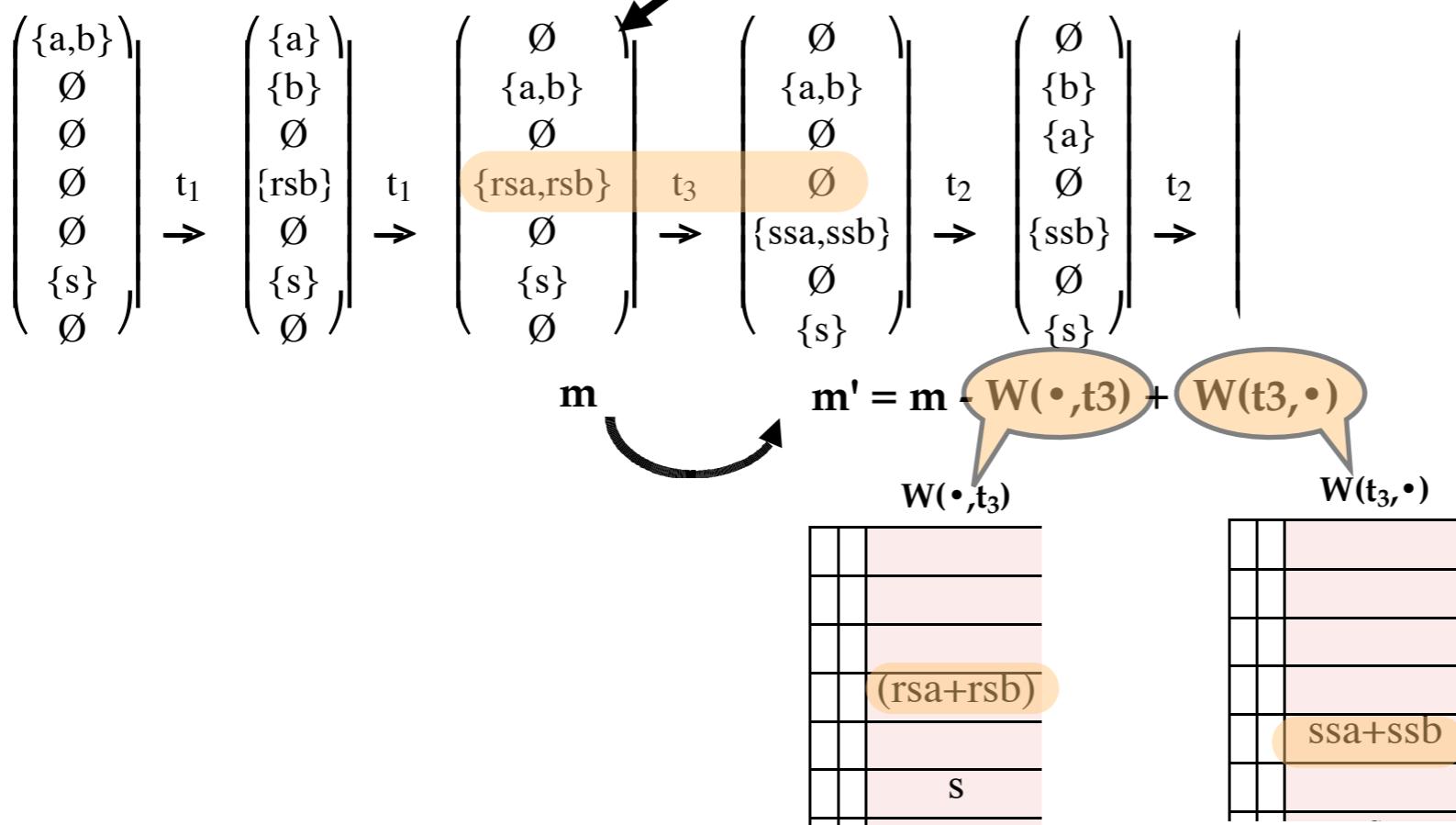
$m' = m - W(\bullet, t_3) + W(t_3, \bullet)$

	t_1	t_2	t_3
p_1			
p_2			
p_3			
p_4		2	
p_5			
p_6			1
p_7			

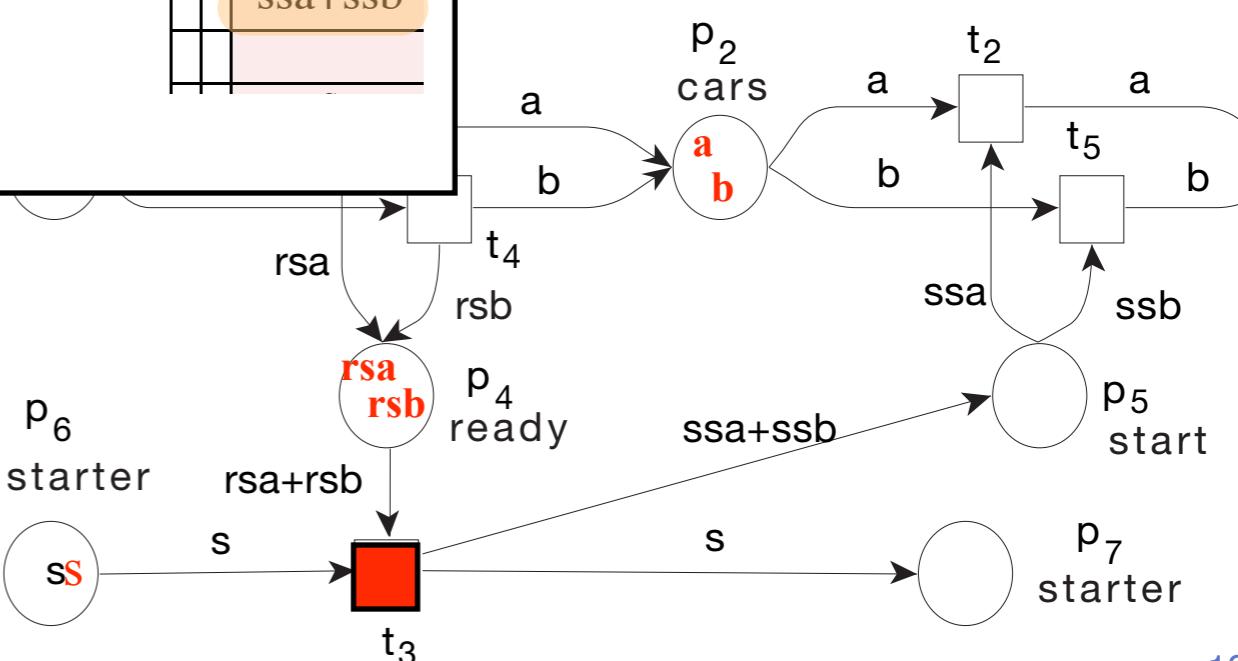
	t_1	t_2	t_3
p_1			
p_2			
p_3			
p_4		2	
p_5			
p_6			1
p_7			

$m \geq W(\cdot, t_3)$

Pre	t ₁	t ₂	t ₃	t ₄
p ₁ :				
p ₂ :				
p ₃ :				
p ₄ :			rsa+rsb	
p ₅ :				
p ₆ :			s	



12



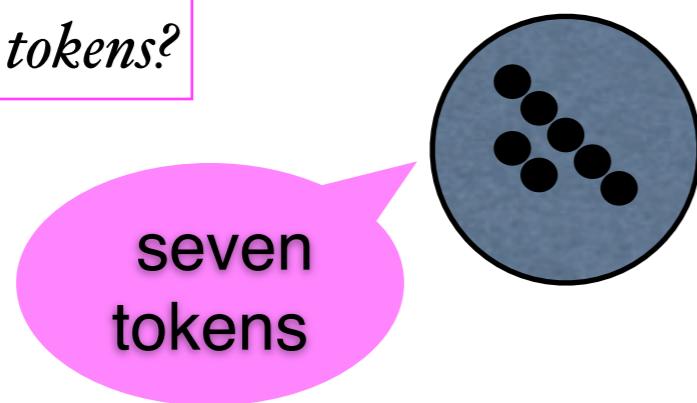
Definition (I) Ein kantenkonstantes Petrinetz (Def. 8.3) $\mathcal{N} = \langle P, T, F, \mathcal{C}, cd, W, \mathbf{m}_0 \rangle$ heißt Platz/Transitions-Netz (P/T -Netz) oder Stellen/Transitions-Netz (S/T -Netz), falls $\mathcal{C} = \{\text{token}\} = \{\{\bullet\}\}$.

$$7' \bullet \quad \cong \quad 7 \in \mathbb{N}$$

$$7' \bullet \stackrel{?}{\equiv} 7 \in \mathbb{N}$$

P/T - net

how many tokens?



seven
tokens

colour set: {}

arc-constant net

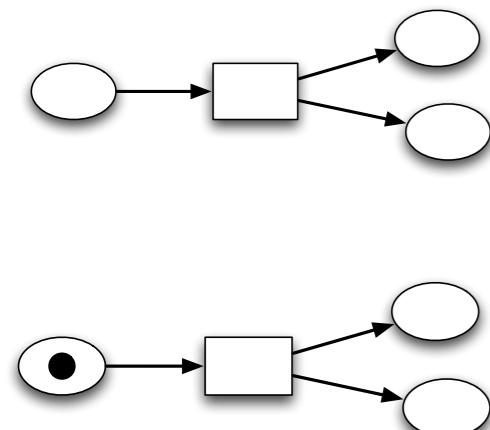


one token

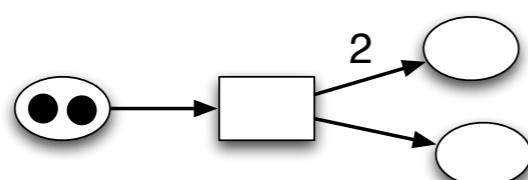
colour set: \mathbb{N}

Netzklassen

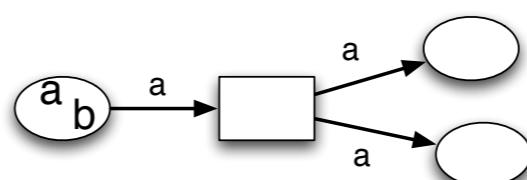
„einfache“ Netze



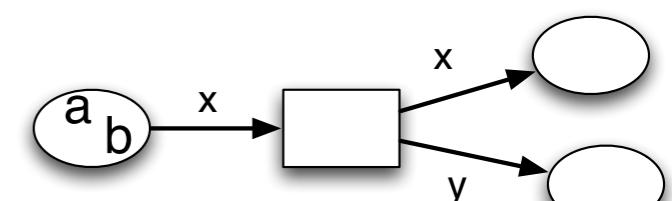
P/T-Netze



kantenkonstante Netze



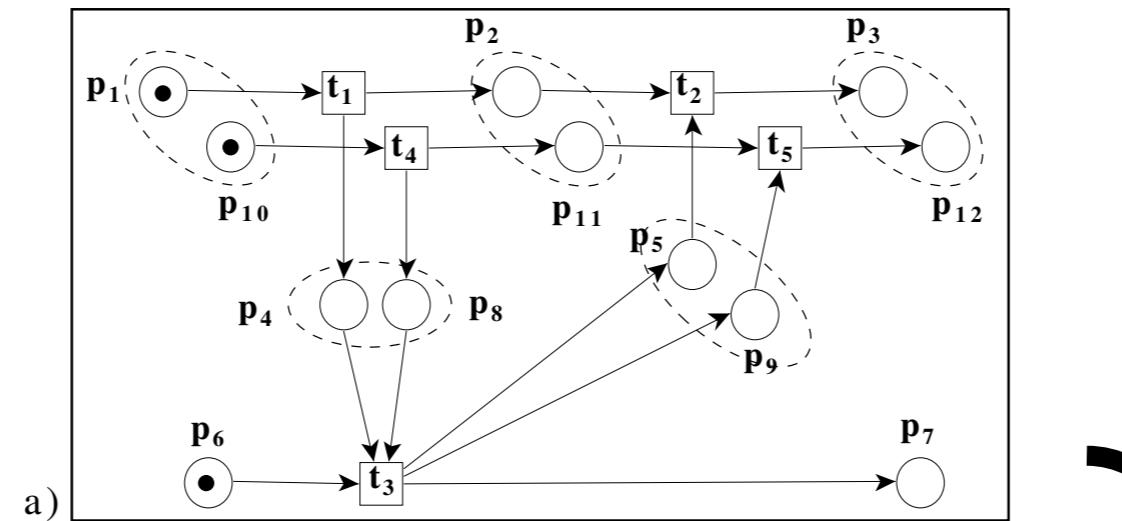
gefärbte Netze



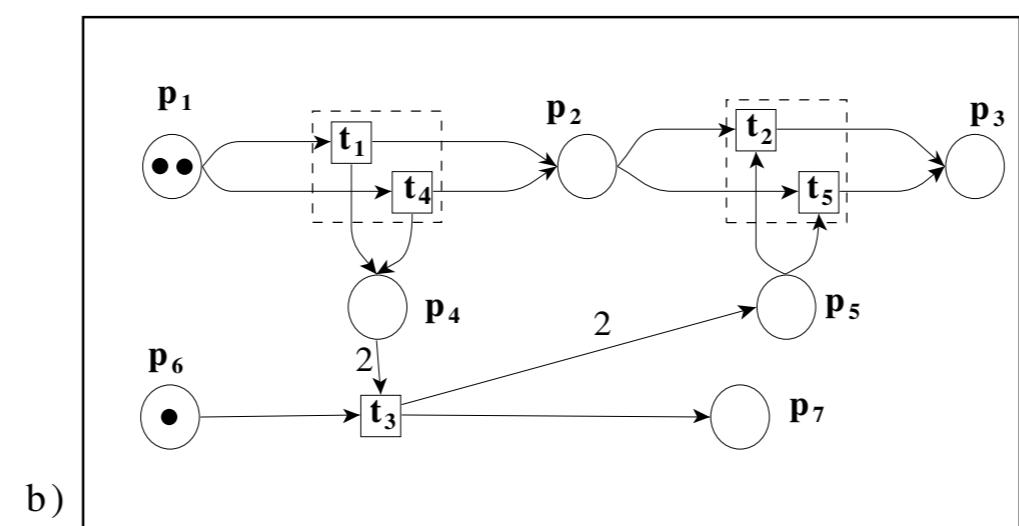
FGI 2

Daniel Moldt

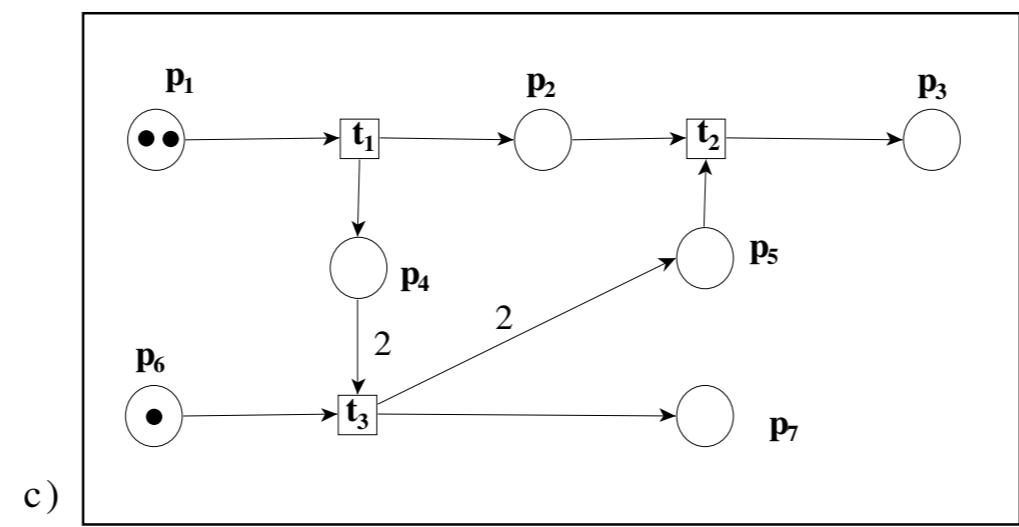
CPN

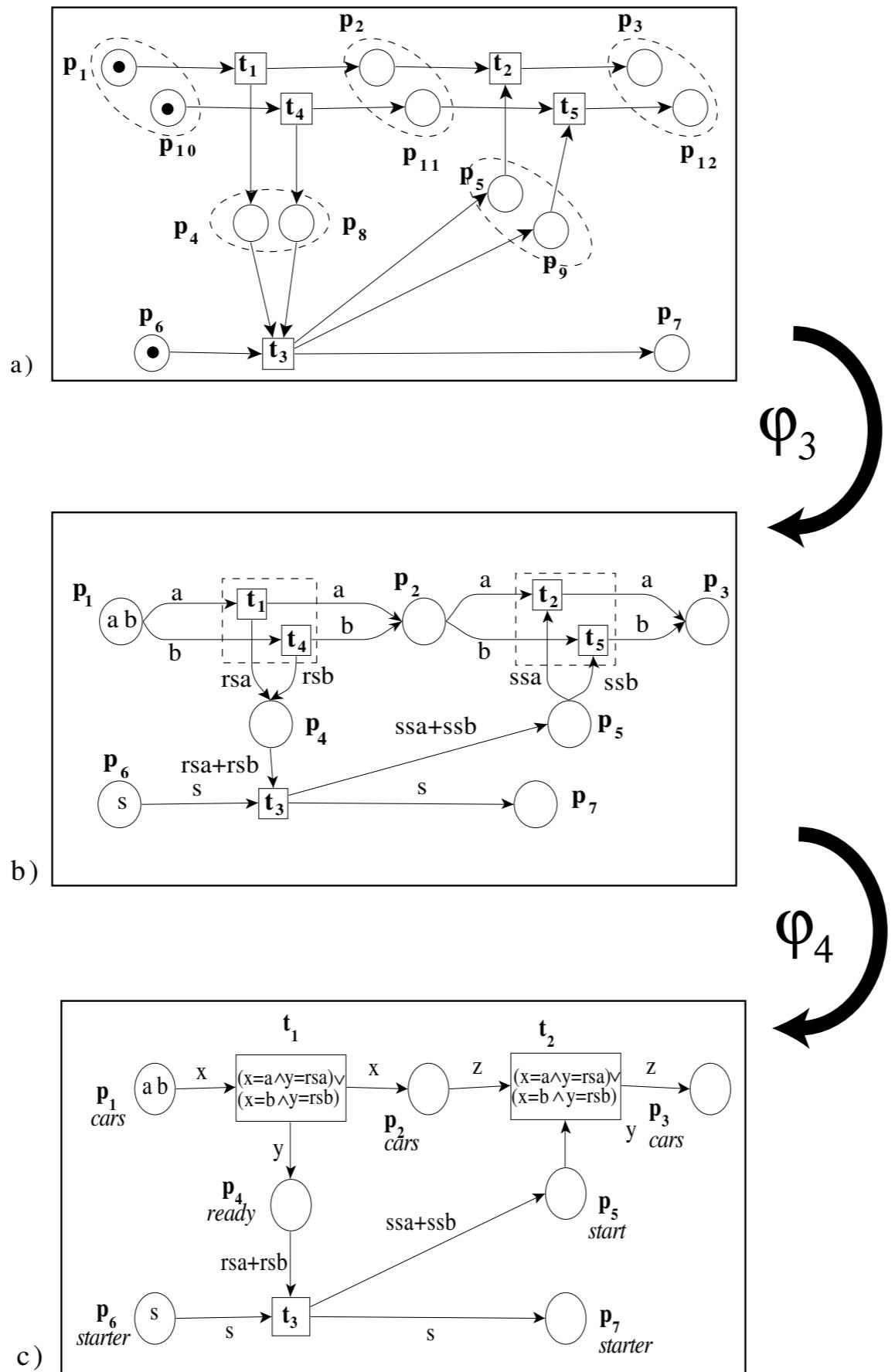


φ_1

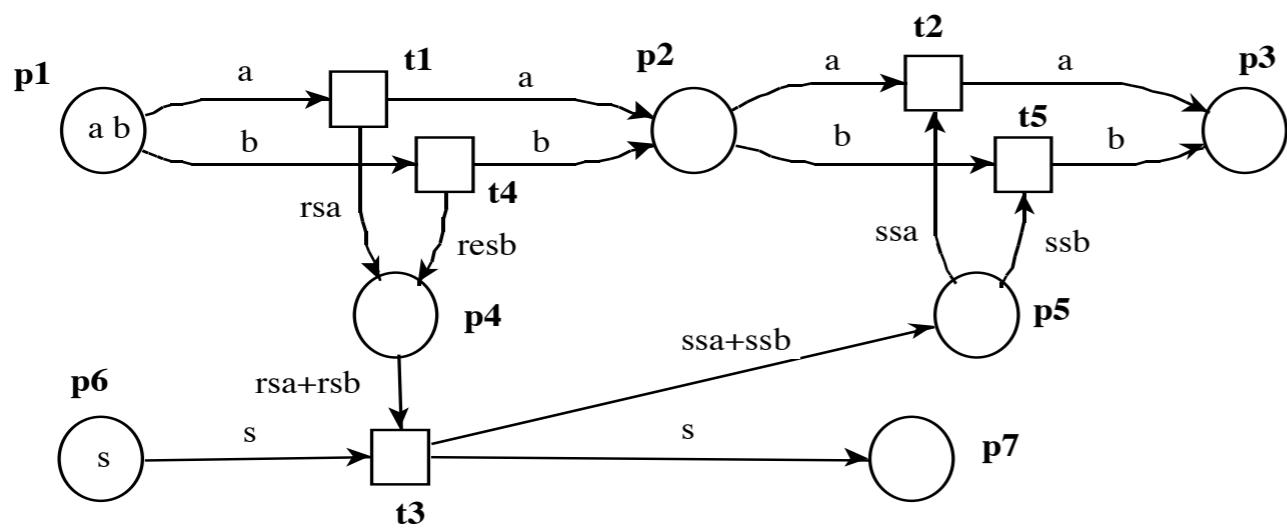


φ_2

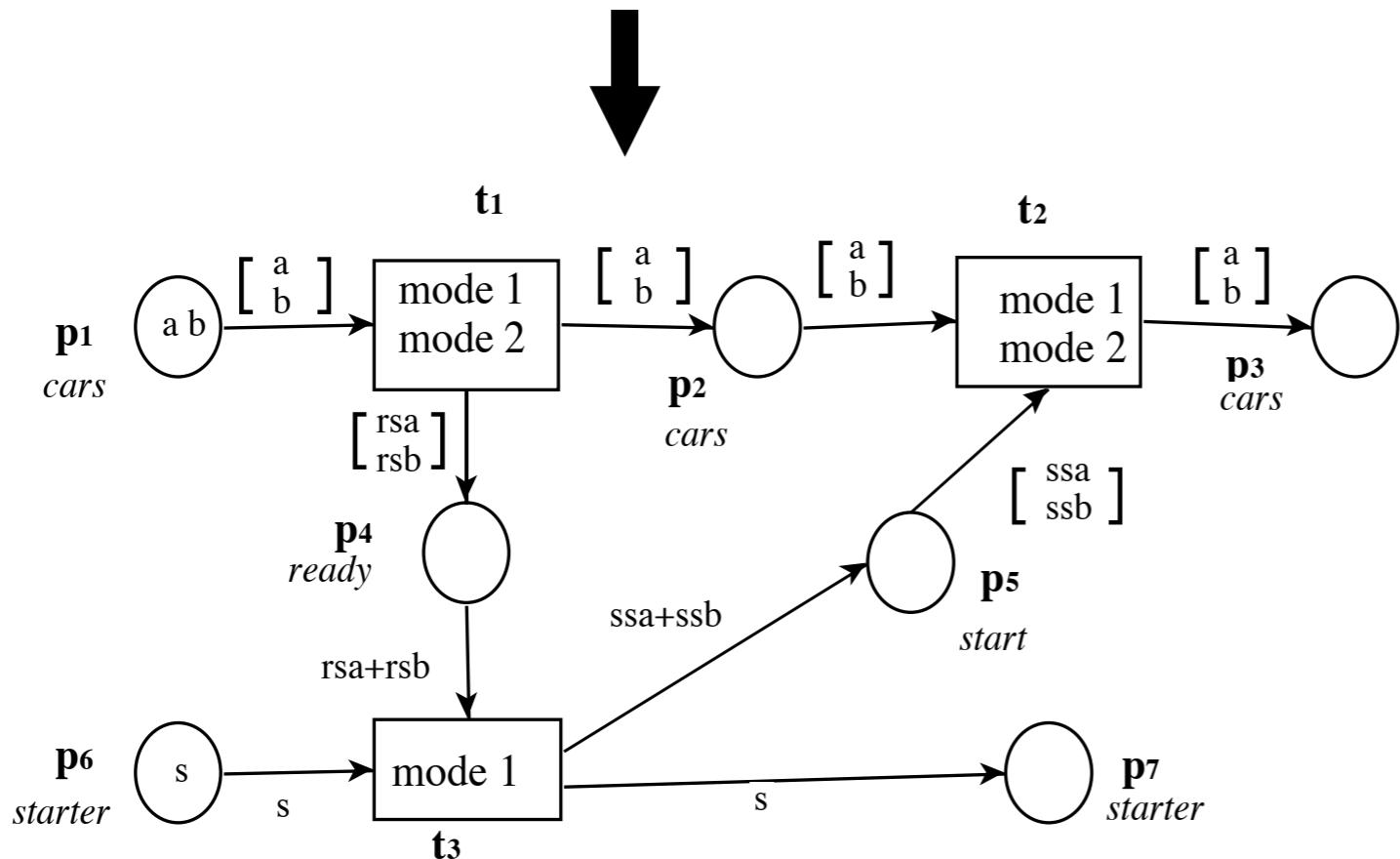




informelle Einführung: gefärbte Netze

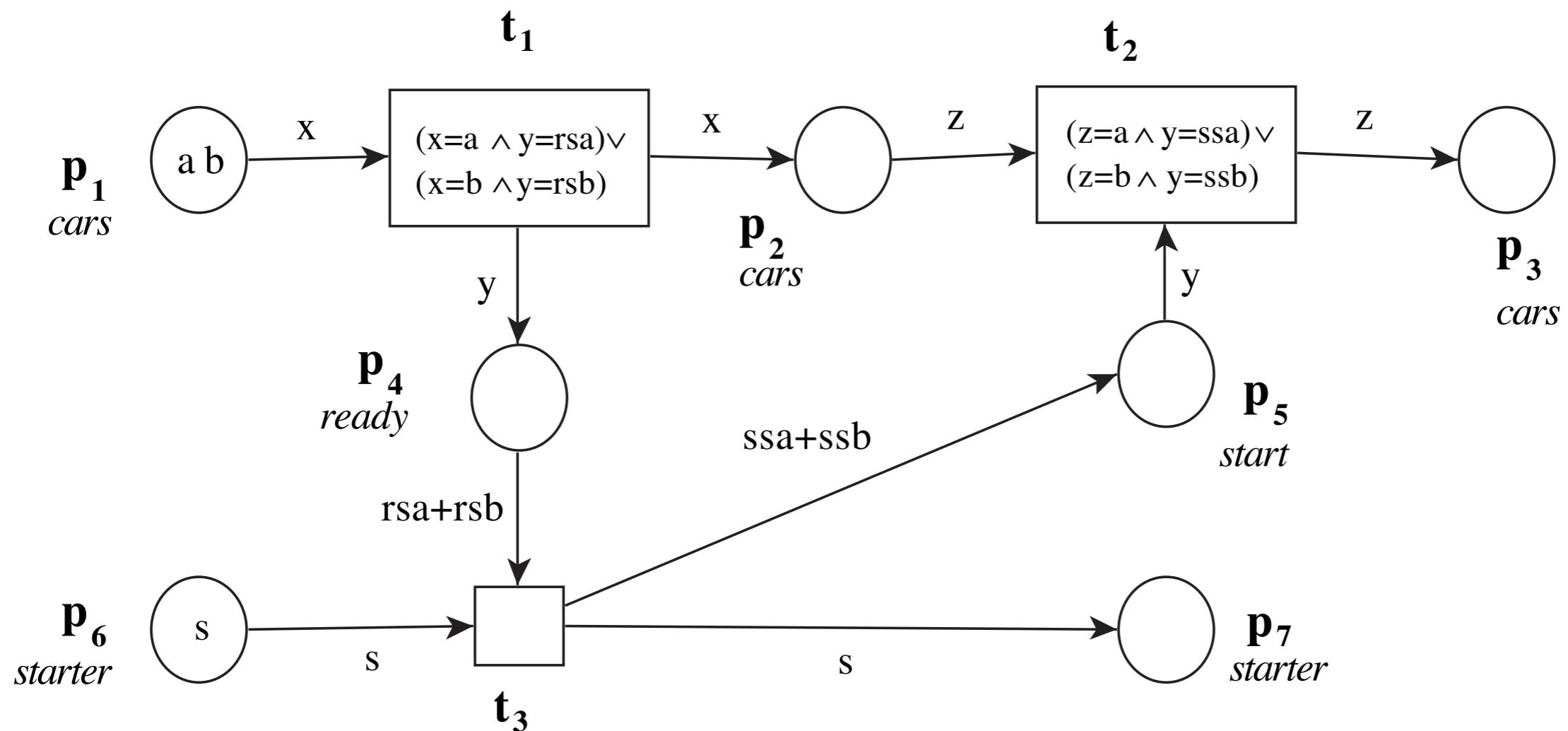


kantenkonstantes
gefäßtes Netz



gefäßtes Netz
mit
Transitions-Modi

Farben:
 $cars = \{a, b\}$
 $starter = \{s\}$
 $ready = \{rsa, rsb\}$
 "ready signs"
 $start = \{ssa, ssb\}$
 "start signals"



colour sets:

cars = {a,b}

starter = {s}

ready = {rsa, rsb} "ready signs"

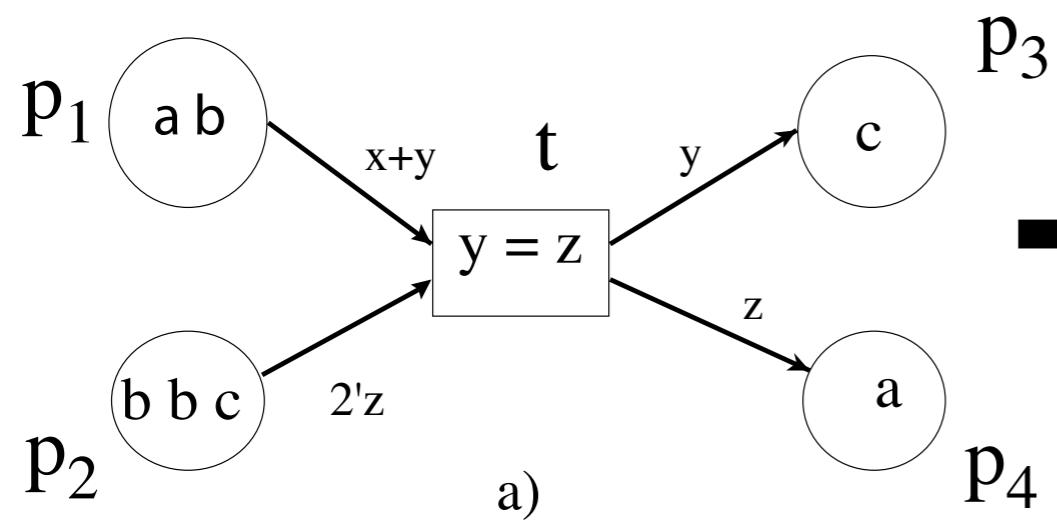
start = {ssa, ssb} "start signs"

variables: x,y,z,s

constants: rsa,rsb,ssa,ssb

Abbildung

Gefärbtes Netz \mathcal{N}_5 mit Guards

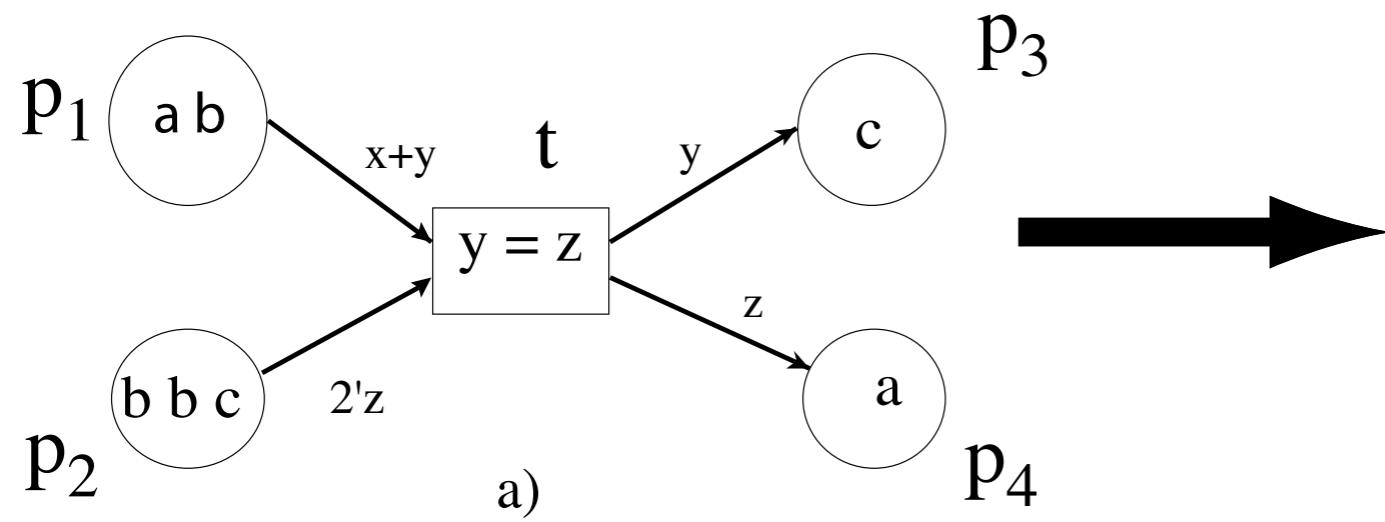


Variablen-Bindung

*Schalten des
kantenkonstanten
Netzes*

*kantenkonstantes
Netz*

Abbildung 8.9 : Schaltregel für gefärbte Netze



wähle eine Belegung, die guard_t erfüllt,
z.B.: $x = a,$
 $y = b,$
 $z = b$

b)

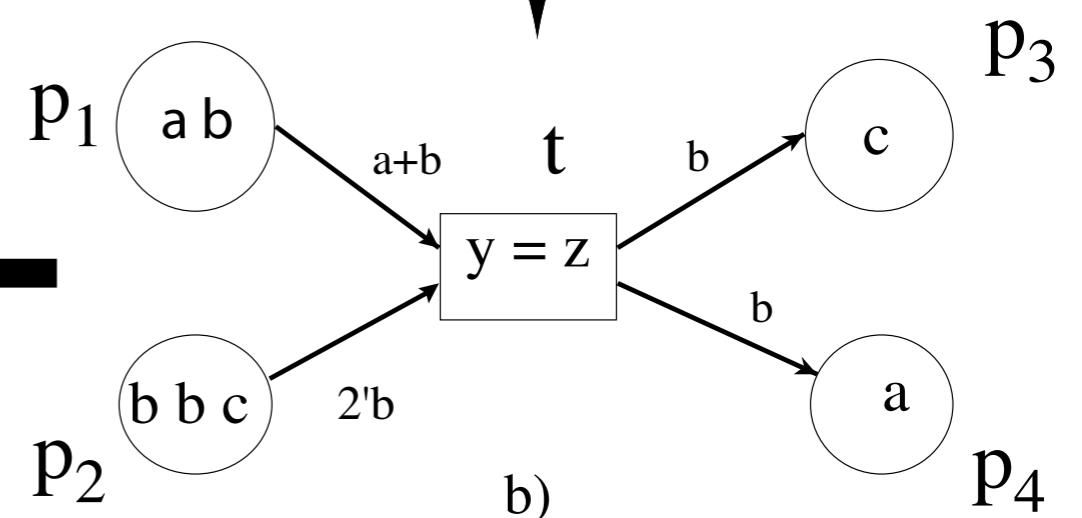
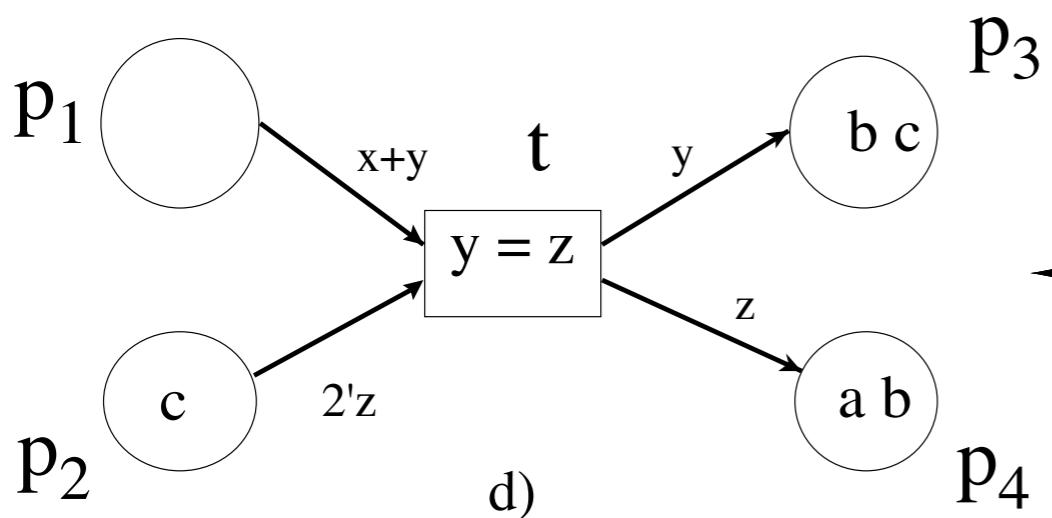
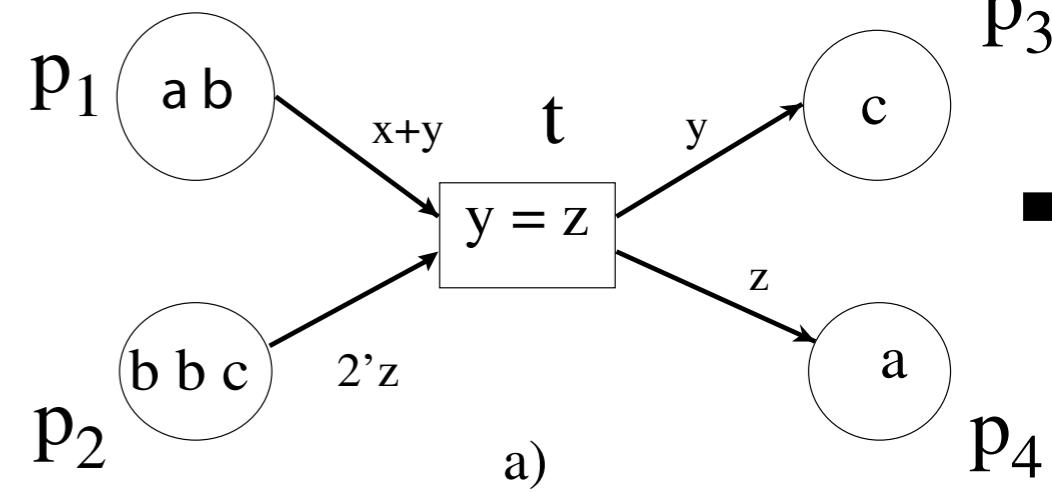


Abbildung 8.9 : Schaltregel für gefärbte Netze

Definition 8.9 Ein gefärbtes Petrinetz (coloured Petri net, CPN) wird als Tupel $\mathcal{N} = \langle P, T, F, \mathcal{C}, cd, Var, Guards, \widehat{W}, \mathbf{m}_0 \rangle$ definiert, wobei gilt:

- (P, T, F) ist ein endliches Netz (Def. 4.1),
- \mathcal{C} ist eine Menge von Farbenmengen,
- $cd: P \rightarrow \mathcal{C}$ ist die Farbzueisungsabbildung (colour domain mapping). Sie wird durch $cd: F \rightarrow \mathcal{C}$, $cd(x, y) := \text{if } x \in P \text{ then } cd(x) \text{ else } cd(y) \text{ fi}$ auf F erweitert.
- Var ist eine Menge von Variablen mit Wertebereichen $\text{dom}(x)$ für jedes $x \in Var$.
- $Guards = \{guard_t \mid t \in T\}$ ordnet jeder Transition $t \in T$ ein Prädikat $guard_t$ mit Variablen aus Var zu.
- $\widehat{W} = \{W_\beta \mid \beta \text{ ist Belegung von } Var\}$ ist eine Menge von Kantengewichtungen der Form $W_\beta : F \rightarrow \text{Bag}(\bigcup \mathcal{C})$, wobei $W_\beta(x, y) \in \text{Bag}(cd(x, y))$ für alle $(x, y) \in F$ gilt.
- $\mathbf{m}_0 : P \rightarrow \text{Bag}(\bigcup \mathcal{C})$ mit $\mathbf{m}_0(p) \in \text{Bag}(cd(p))$ für alle $p \in P$ ist die Anfangsmarkierung.

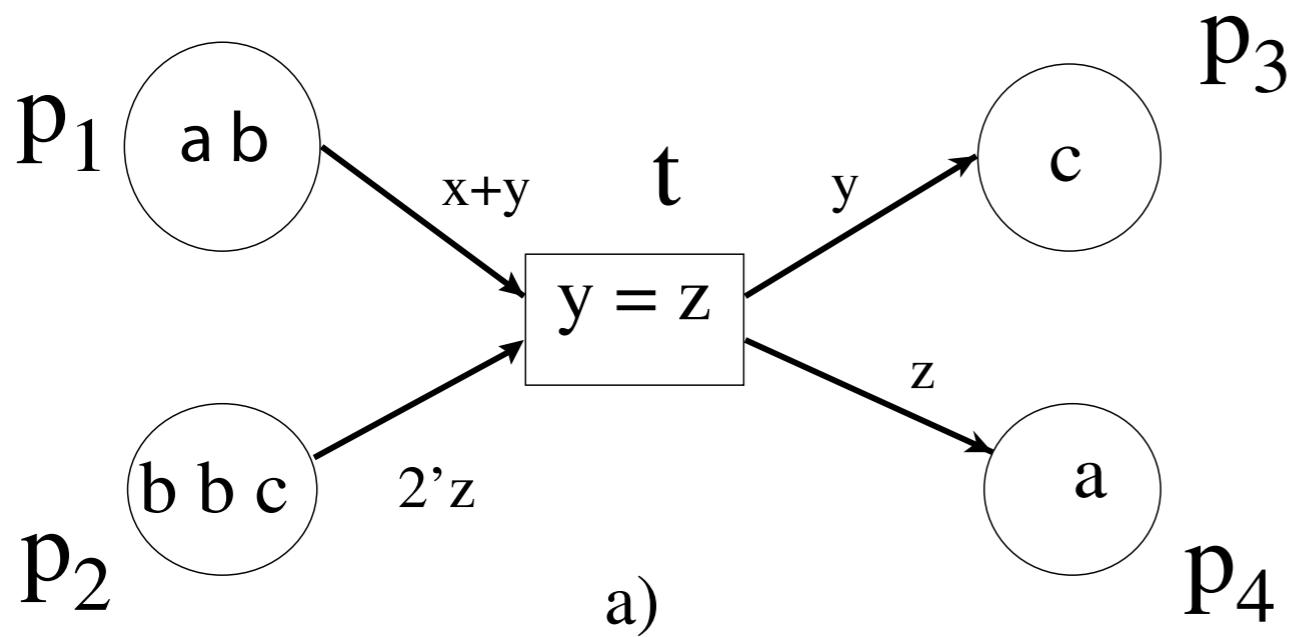


wähle eine Belegung, die $guard_t$ erfüllt,
z.B.: $x = a$, $y = b$, $z = b$

β a
 b β_1
 c

Nicht schaltbar

eine andere Belegung?



wähle eine Belegung, die guard_t erfüllt,

z.B.: $x = a, \quad a$
 $y = b, \quad b \quad \beta_1$
 $z = b \quad c$

Definition 8.10 a) Die Markierung eines gefärbten Netzes (CPN)

$\mathcal{N} = \langle P, T, F, \mathcal{C}, cd, Var, Guards, \widehat{W}, \mathbf{m}_0 \rangle$ ist ein Vektor \mathbf{m} mit $\mathbf{m}(p) \in Bag(cd(p))$ für jedes $p \in P$ (auch als Abbildung $\mathbf{m} : P \rightarrow Bag(\bigcup \mathcal{C})$ mit $\mathbf{m}(p) \in Bag(cd(p))$ für jedes $p \in P$ aufzufassen).

b) Sei β eine Belegung für Var . Die Transition $t \in T$ heißt β -aktiviert in einer Markierung \mathbf{m} falls $\text{guard}_t(\beta) = \text{true}$ und $\forall p \in \bullet t. \mathbf{m}(p) \geq W_\beta(p, t)$ (als Relation: $\mathbf{m} \xrightarrow{t, \beta}$).

c) Es sei

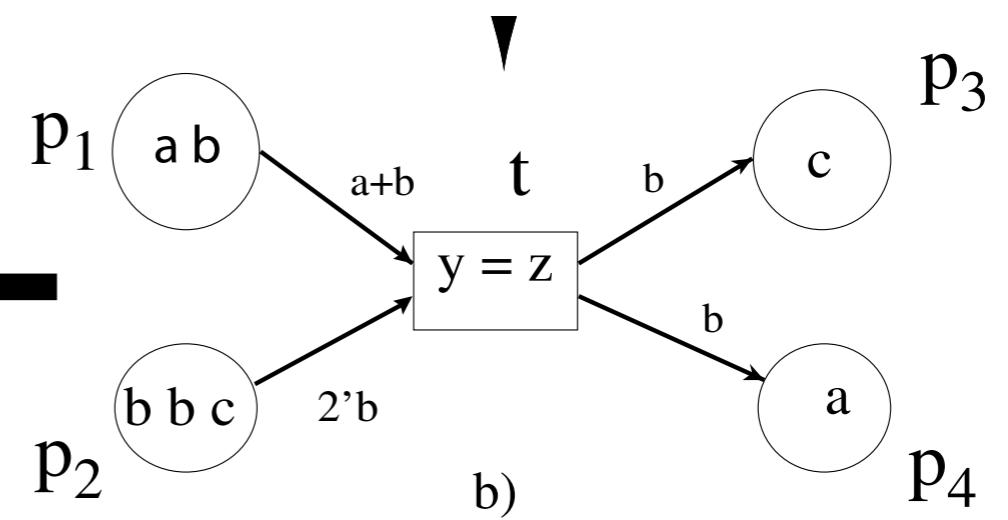
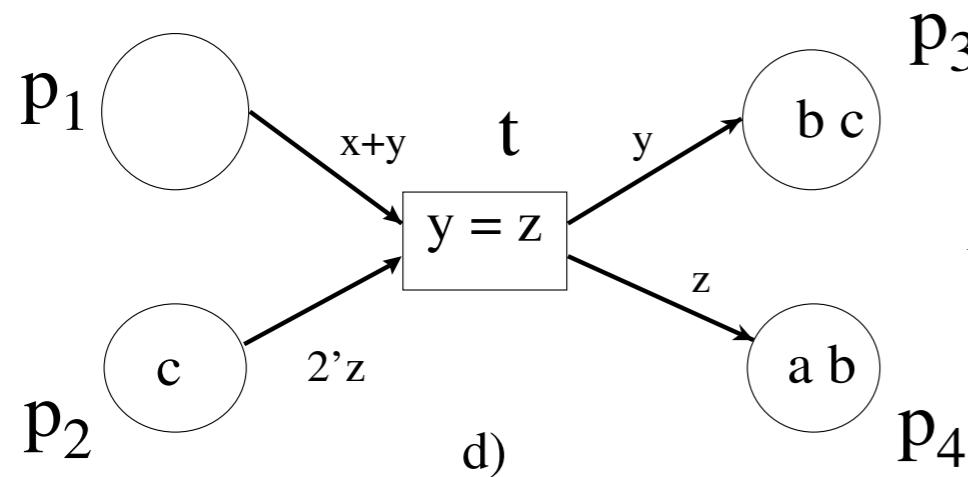
$$\widetilde{W}_\beta(p, t) := \begin{cases} W_\beta(p, t) & \text{falls } (p, t) \in F \\ \emptyset & \text{sonst} \end{cases} \quad \text{und entsprechend}$$

$$\widetilde{W}_\beta(t, p) := \begin{cases} W_\beta(t, p) & \text{falls } (t, p) \in F \\ \emptyset & \text{sonst} \end{cases}.$$

Ist t in \mathbf{m} β -aktiviert, dann ist die Nachfolgemarkierung definiert durch:

$$\mathbf{m} \xrightarrow{t, \beta} \mathbf{m}' \Leftrightarrow \forall p \in P. (\mathbf{m}(p) \geq \widetilde{W}_\beta(p, t) \wedge \mathbf{m}'(p) = \mathbf{m}(p) - \widetilde{W}_\beta(p, t) + \widetilde{W}_\beta(t, p)).$$

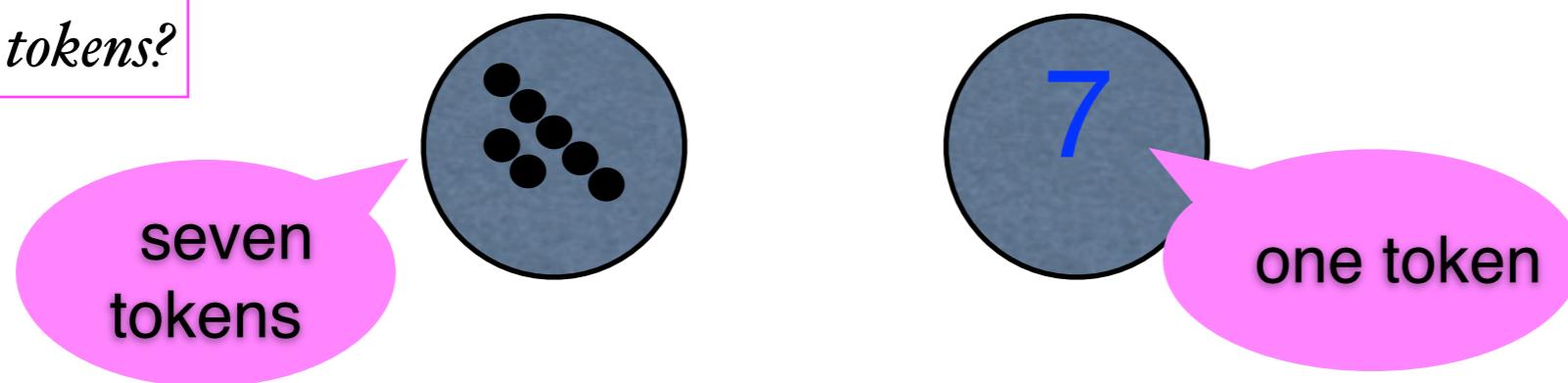
(Beachte, dass es sich um Multimengenoperationen handelt!).



$$7' \bullet \stackrel{?}{\equiv} 7 \in \mathbb{N}$$

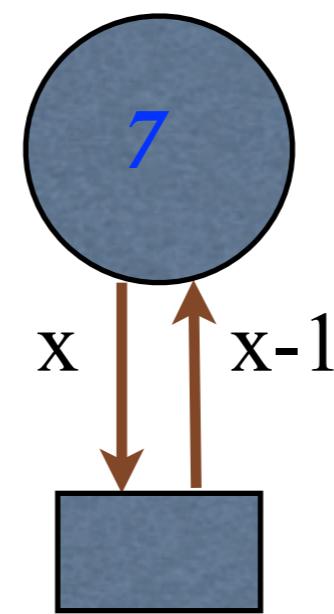
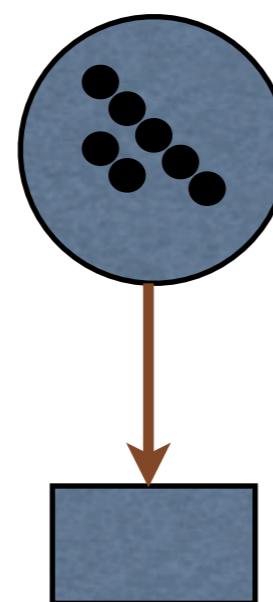
P/T - net

how many tokens?

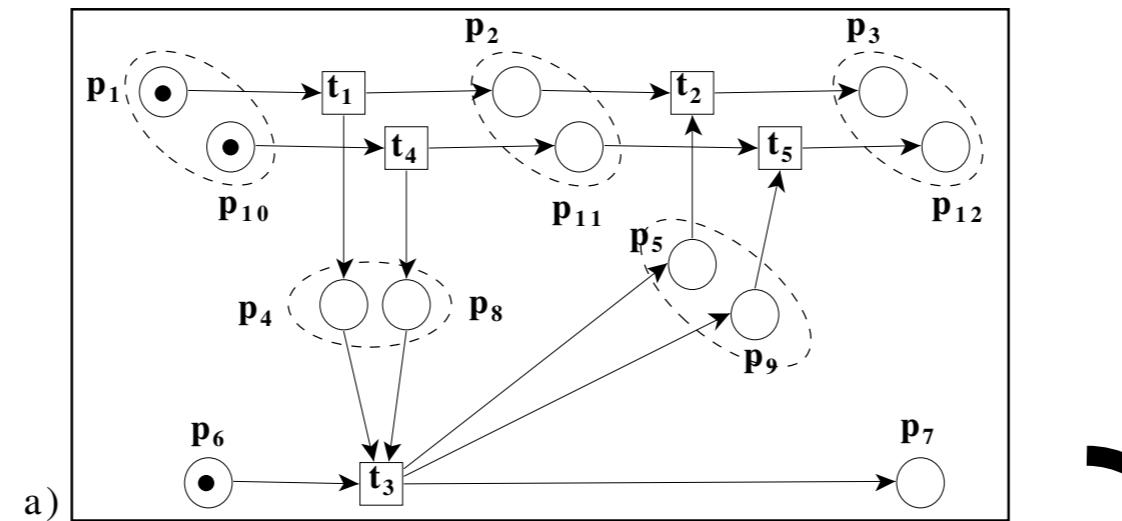


colour set: {●}

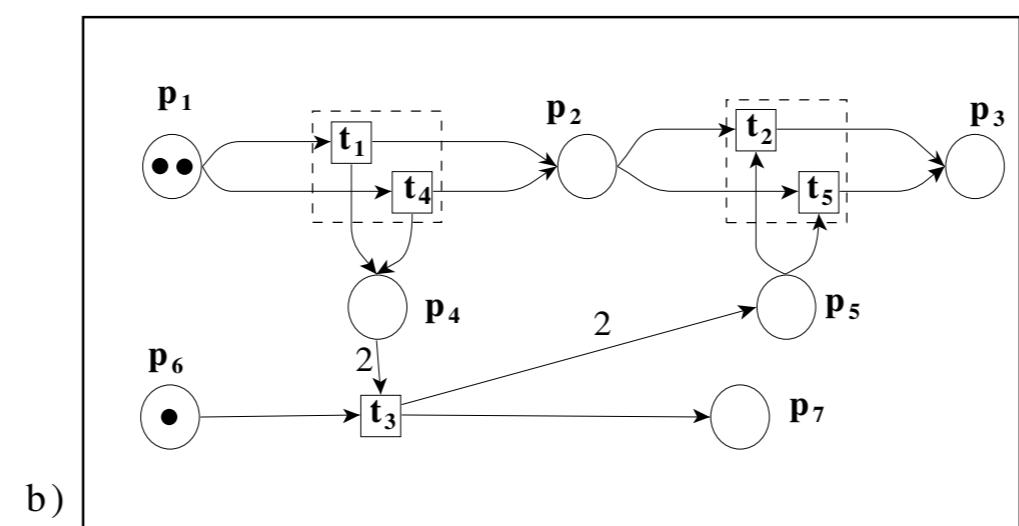
counting down



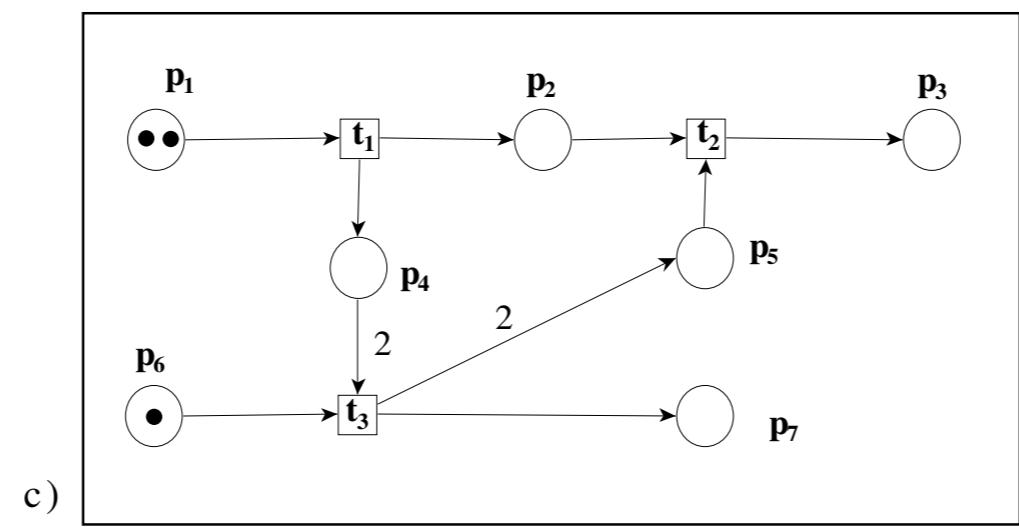
coloured net

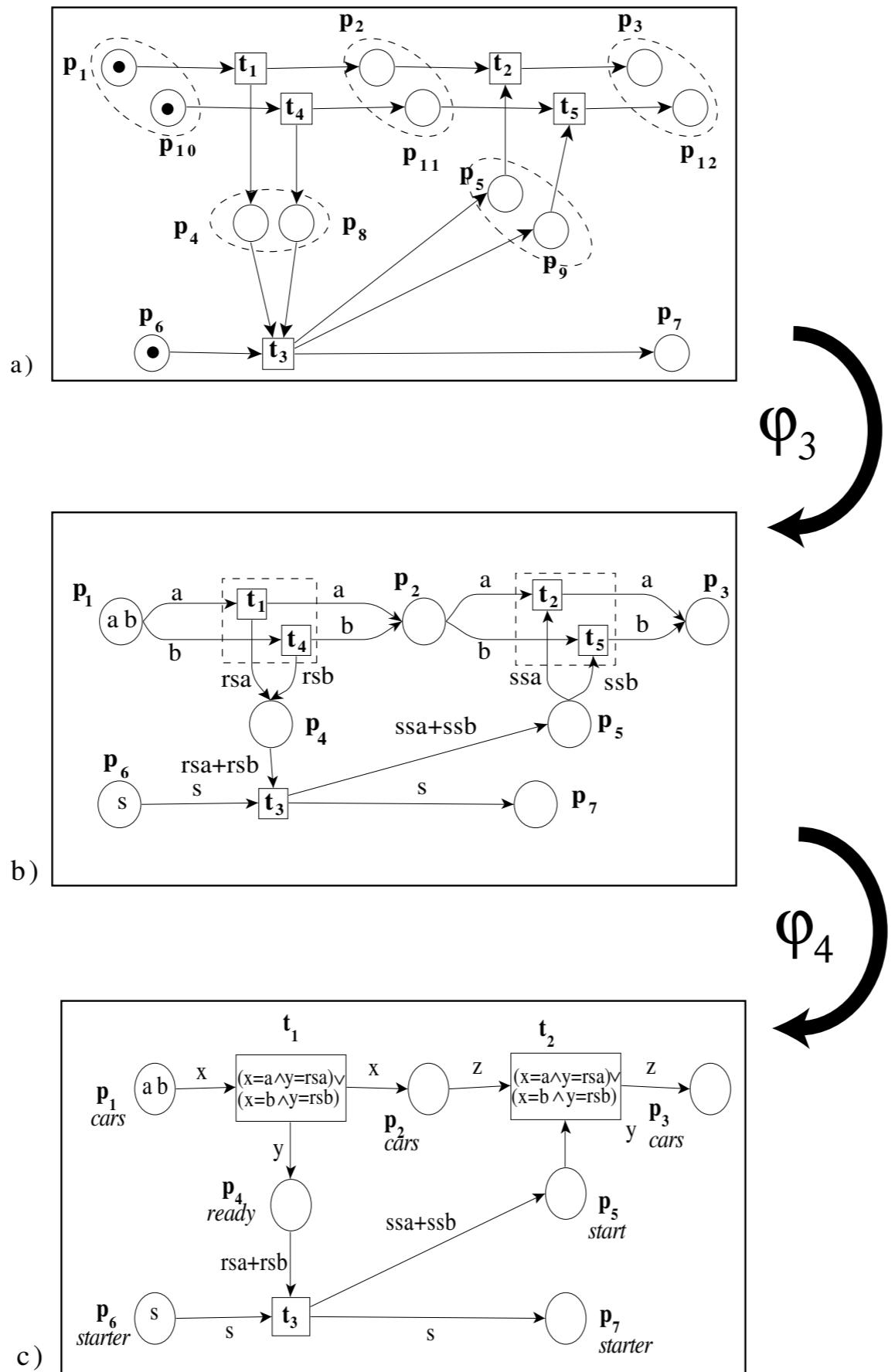


φ_1



φ_2



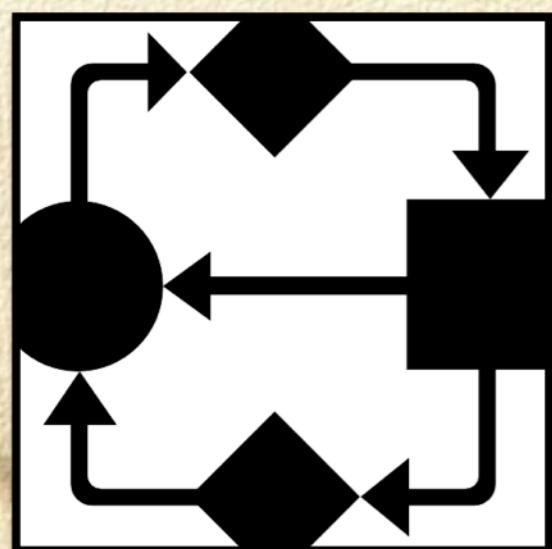
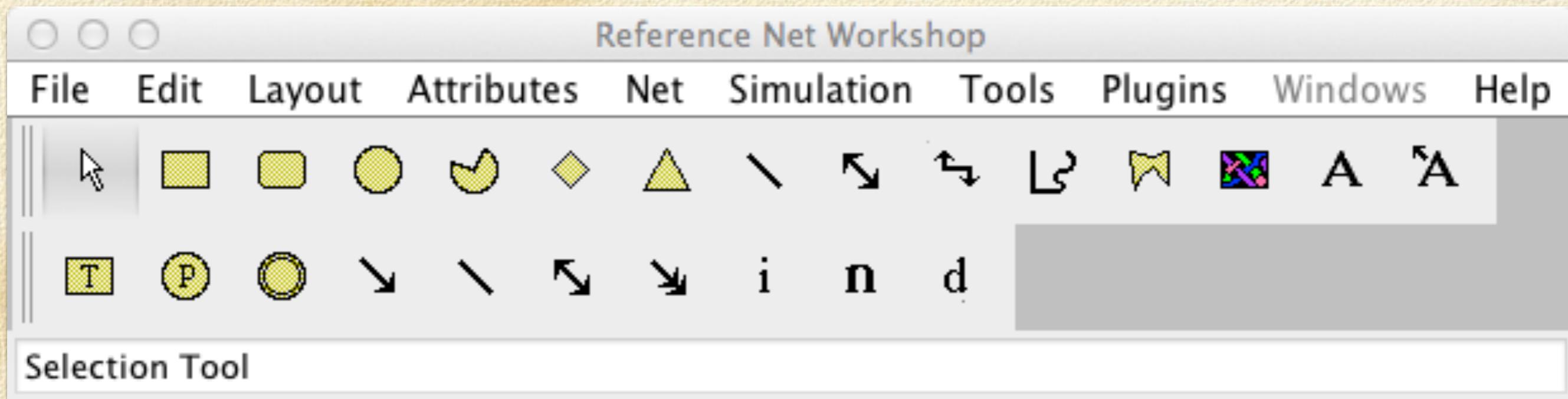


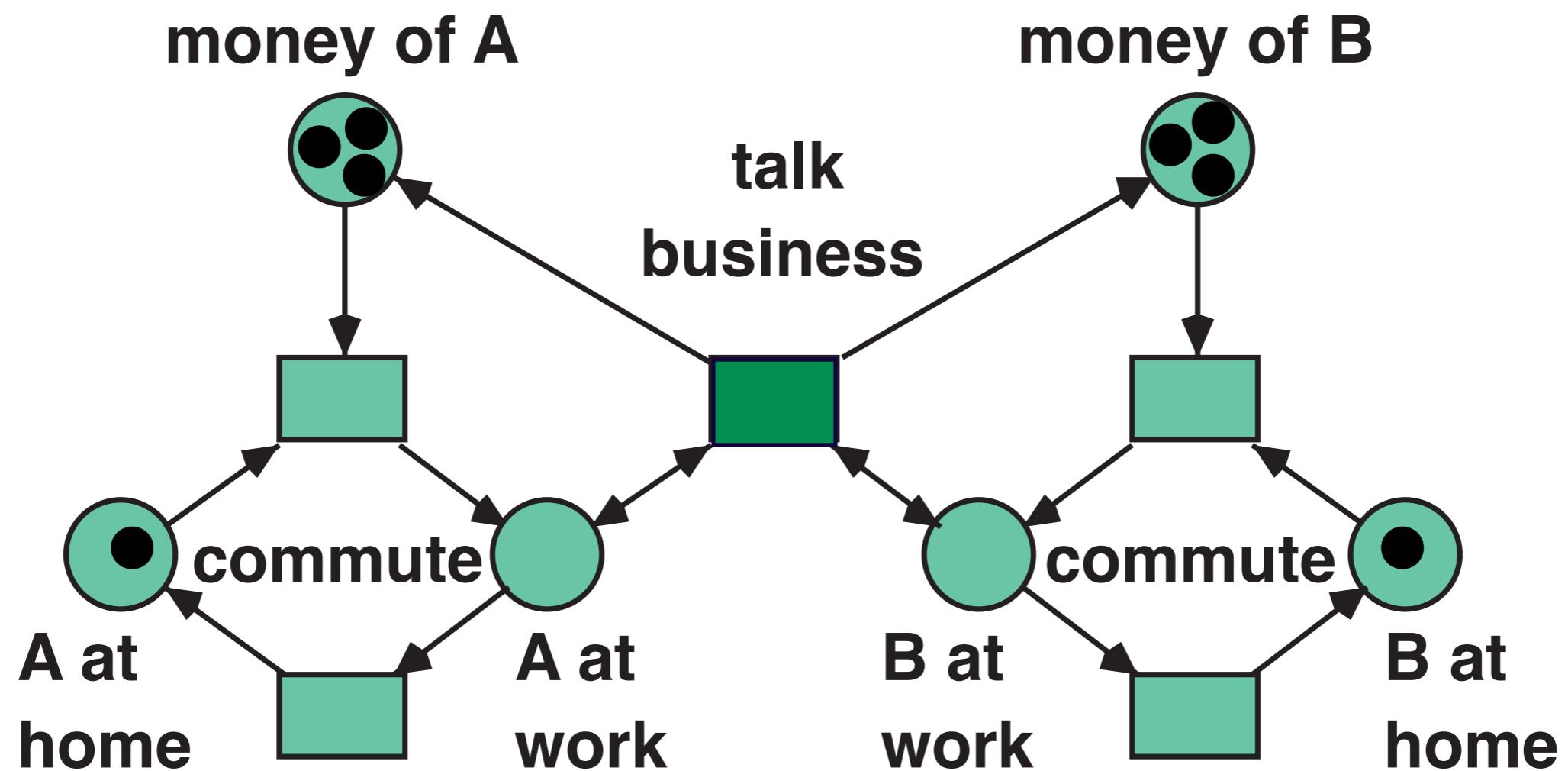
FGI 2

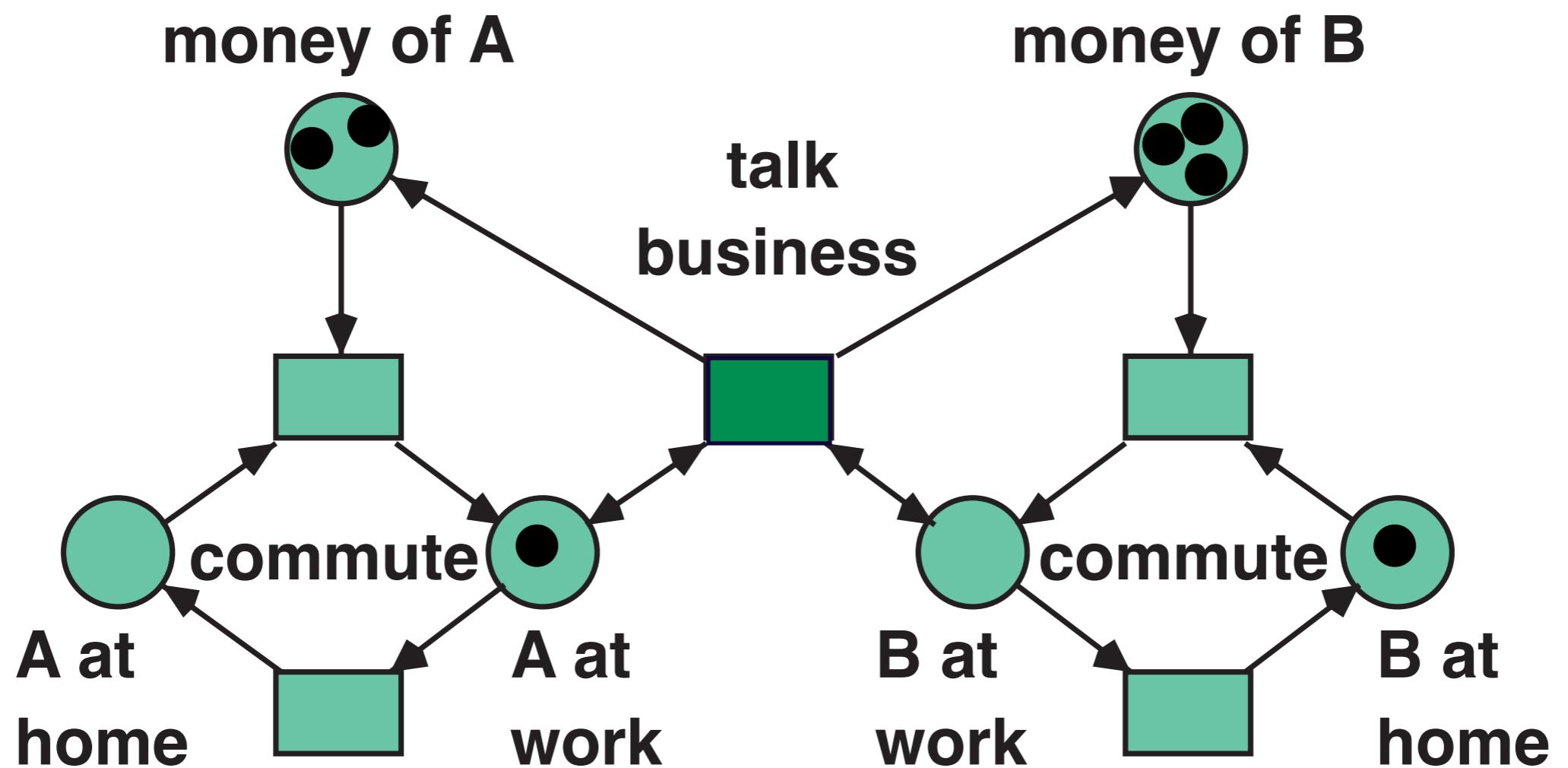
Daniel Moldt

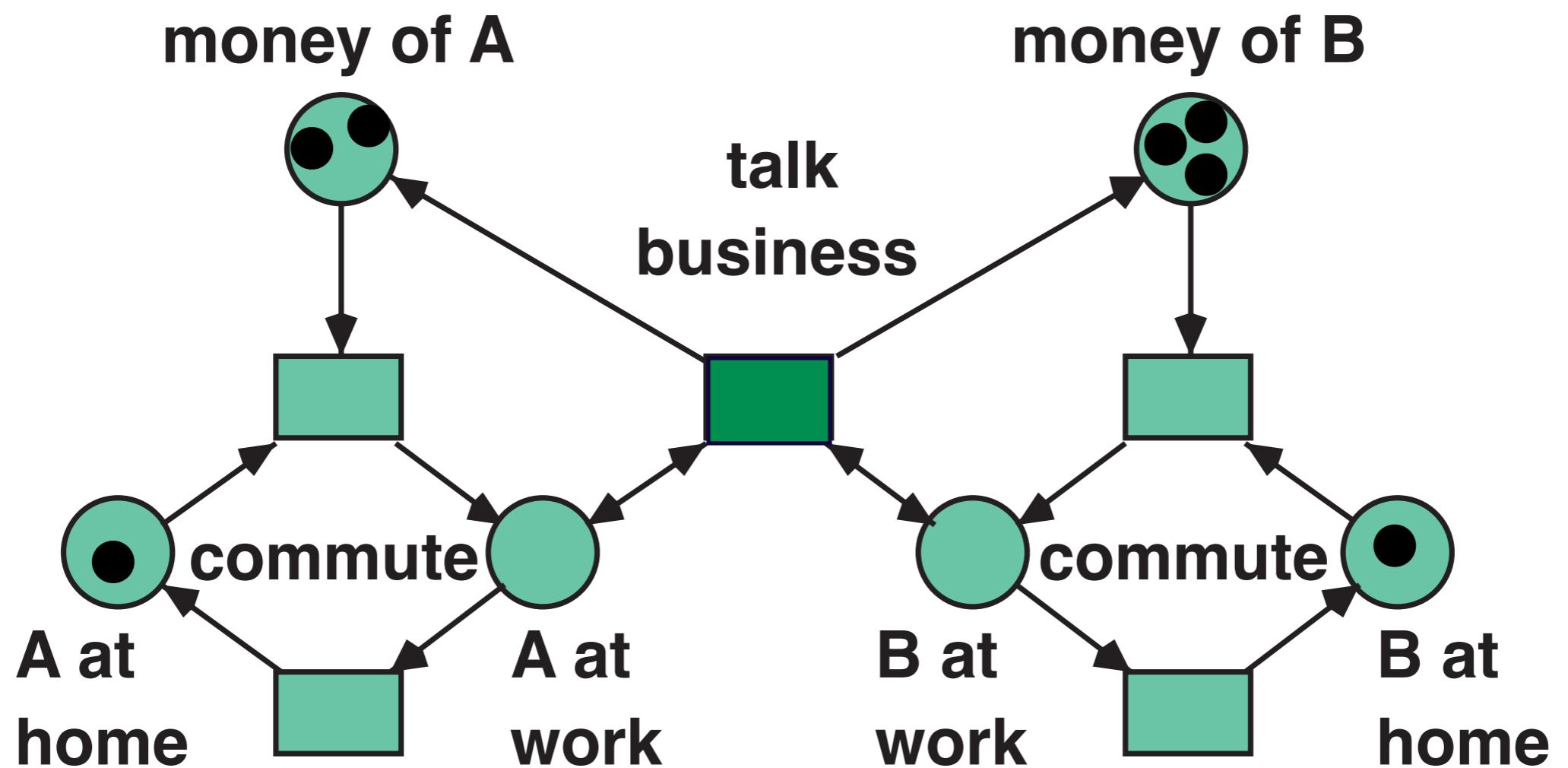
Renew

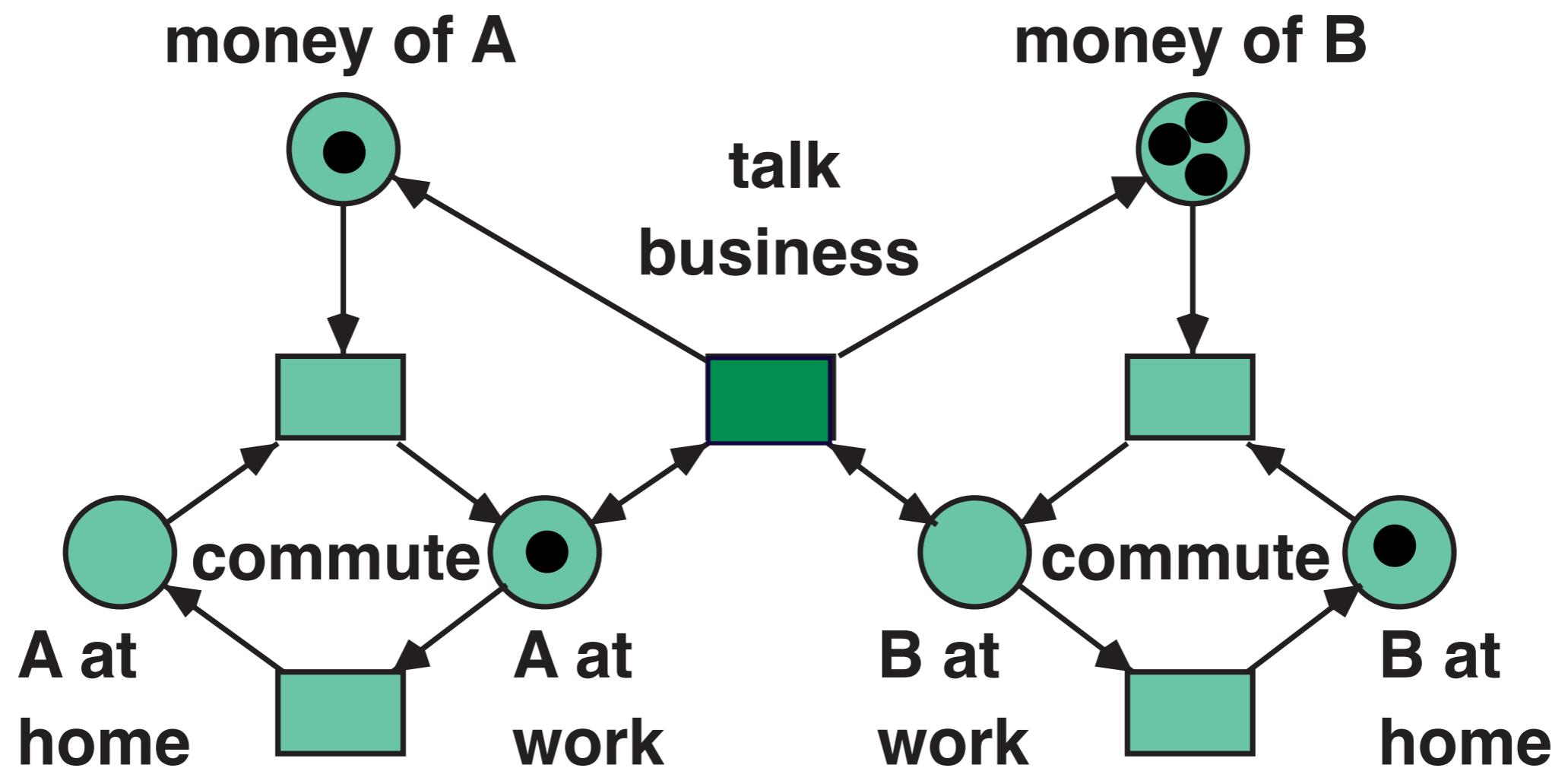
Das RENEW-Werkzeug

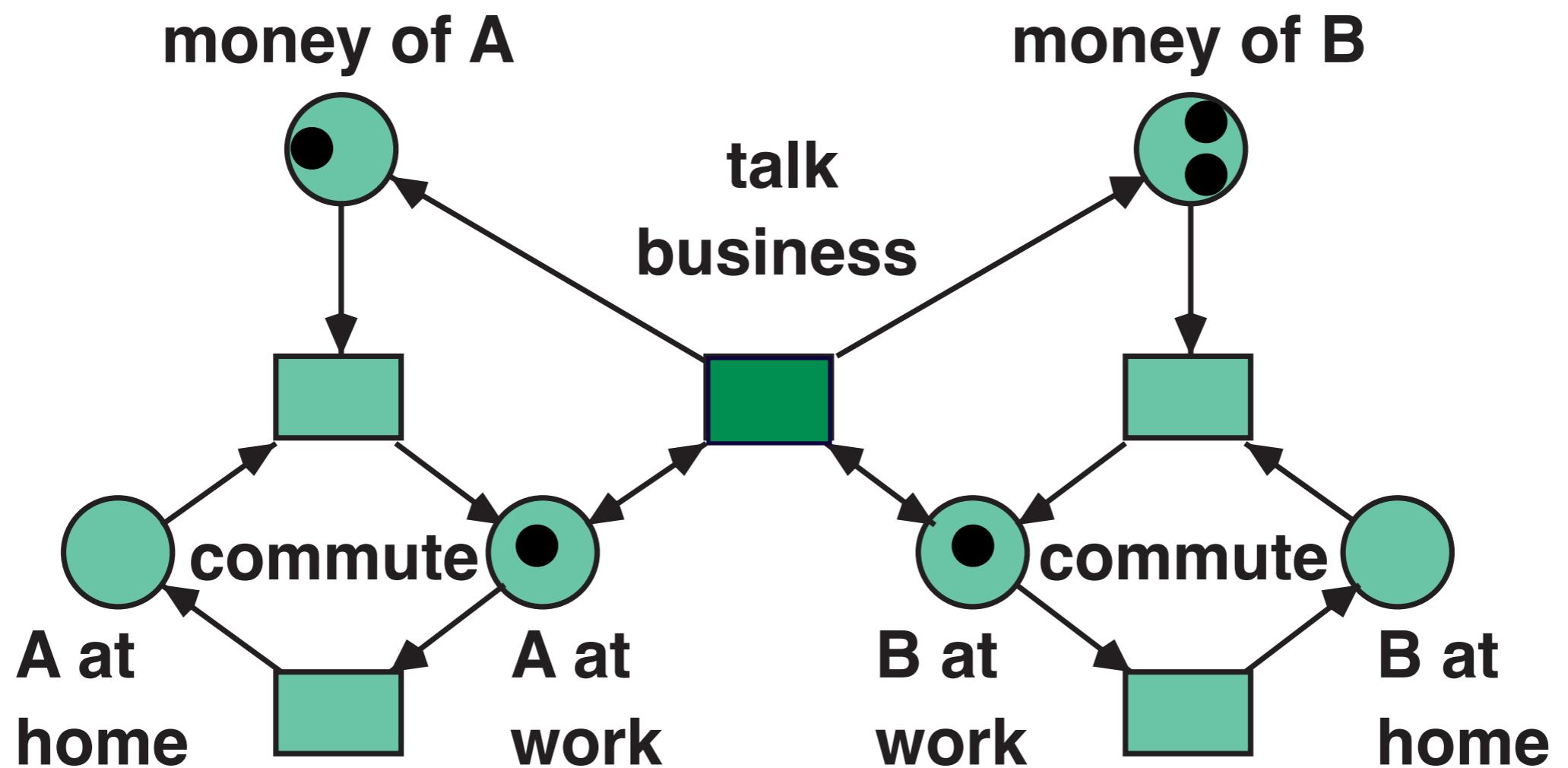


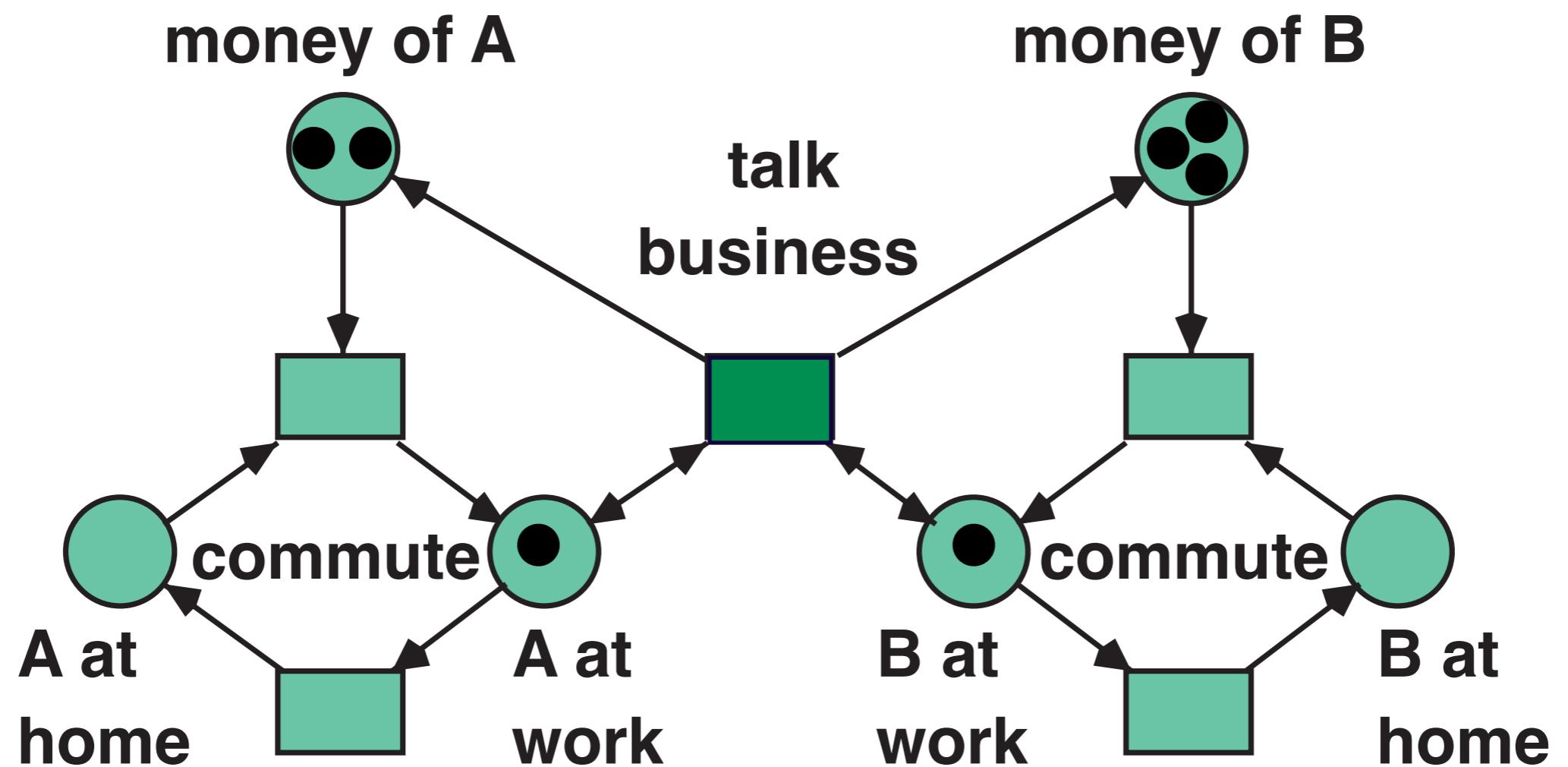


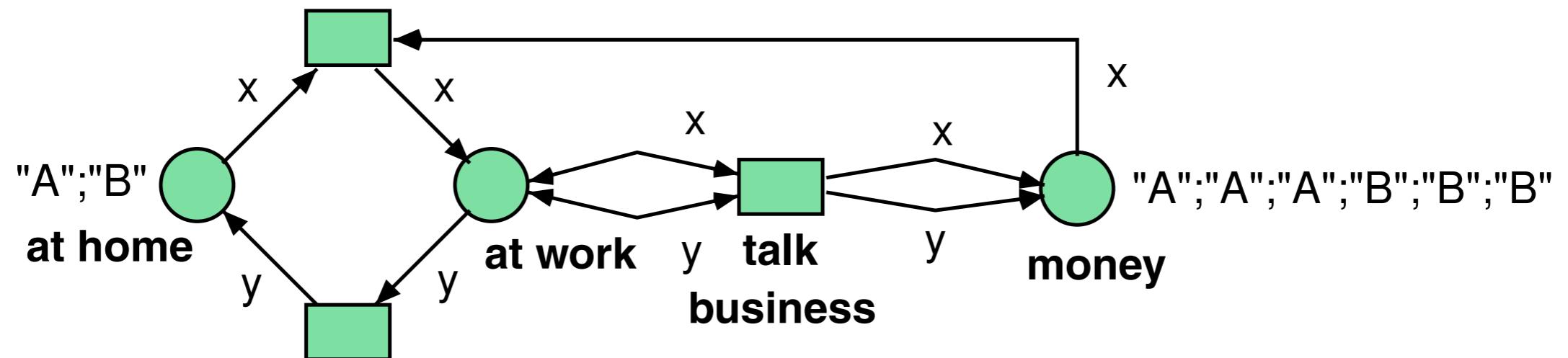












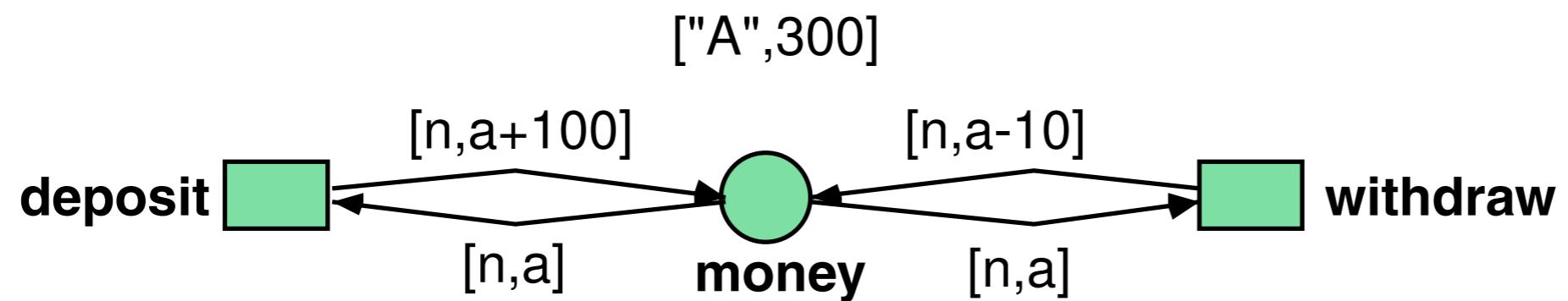


Abbildung 4.13: Ein nicht ganz so einfaches Bankkonto

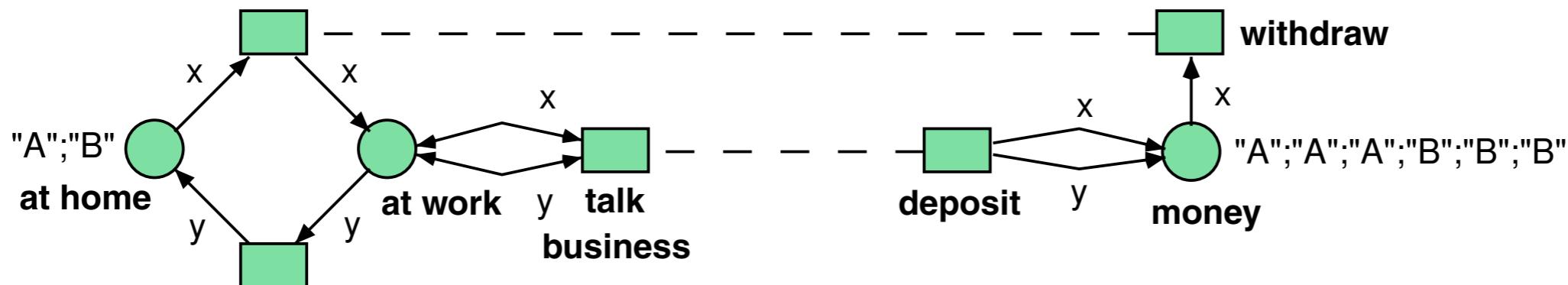
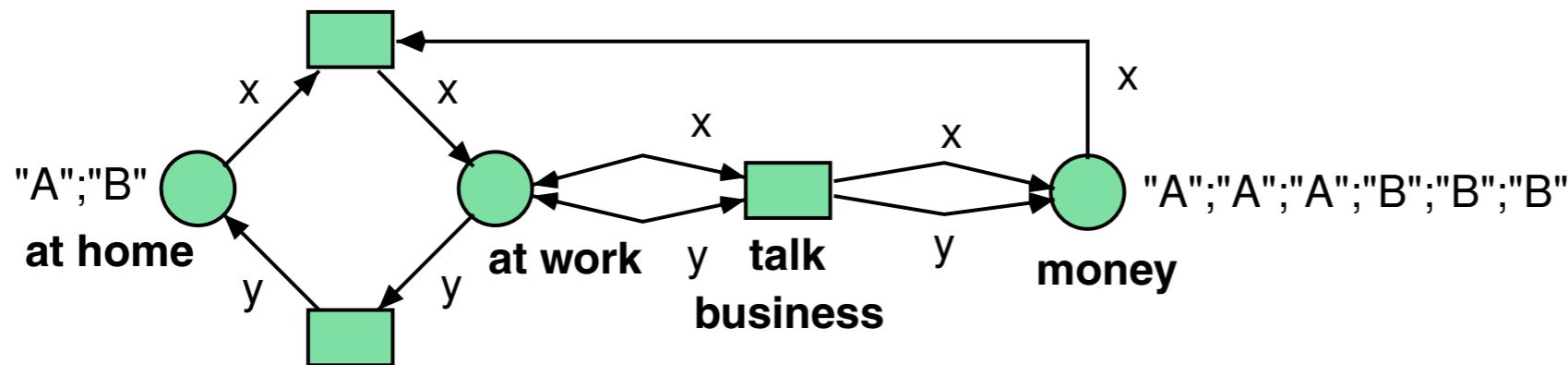


Abbildung 4.26: Personen und Konten werden geteilt



Synchronisation durch Kanäle

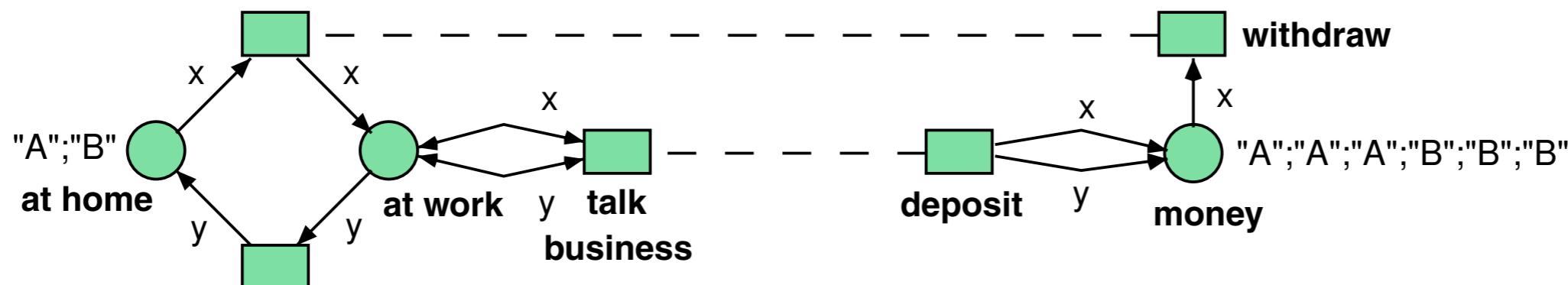


Abbildung 4.26: Personen und Konten werden geteilt

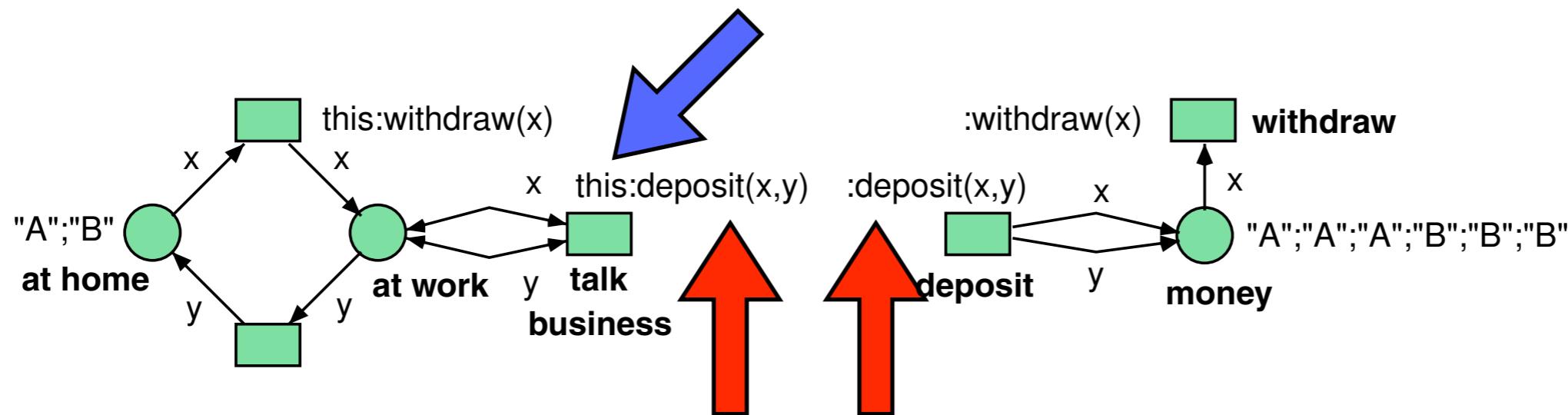


Abbildung 4.27: Textuelle Anschriften kennzeichnen Synchronisation

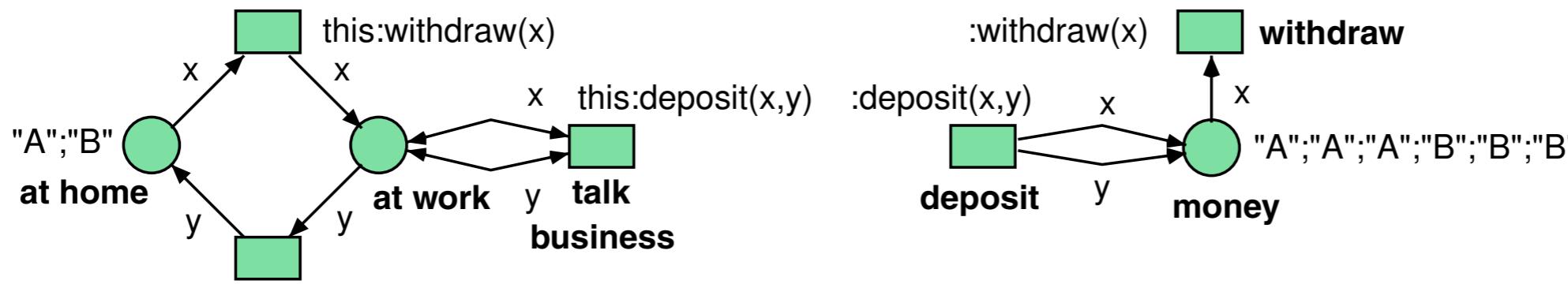


Abbildung 4.27: Textuelle Anschriften kennzeichnen Synchronisation

Bezeichner eines Netzes

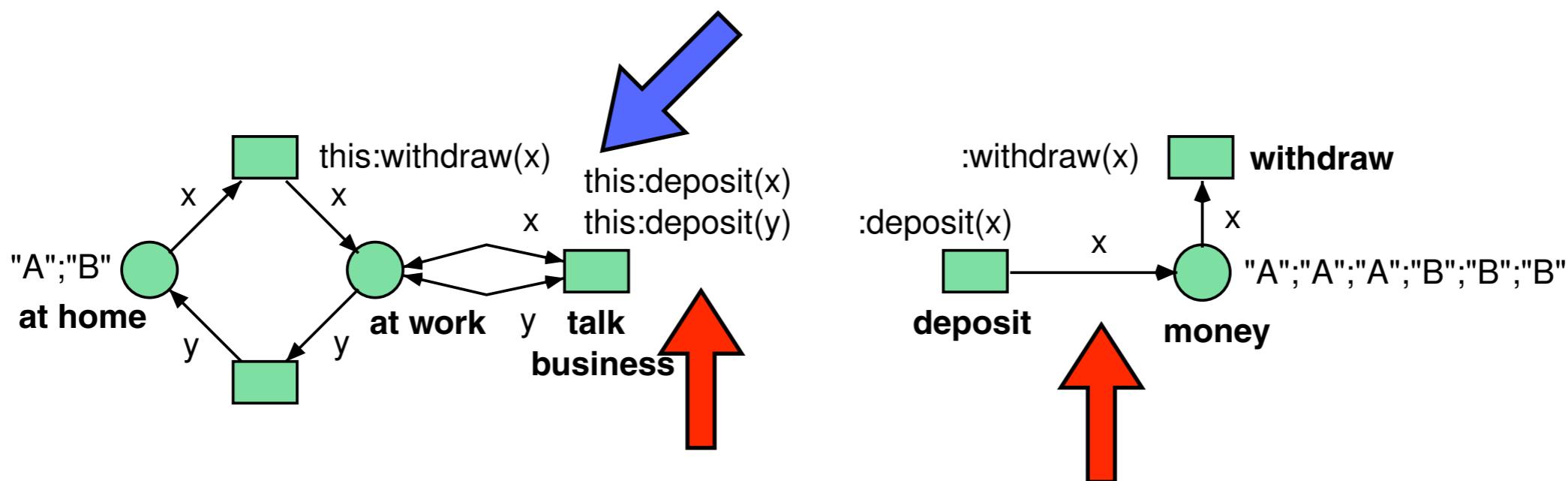
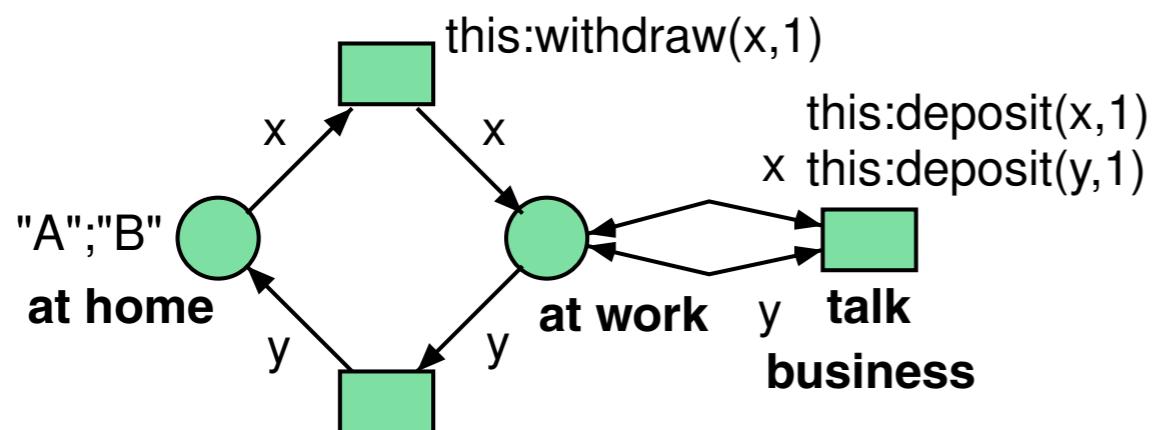
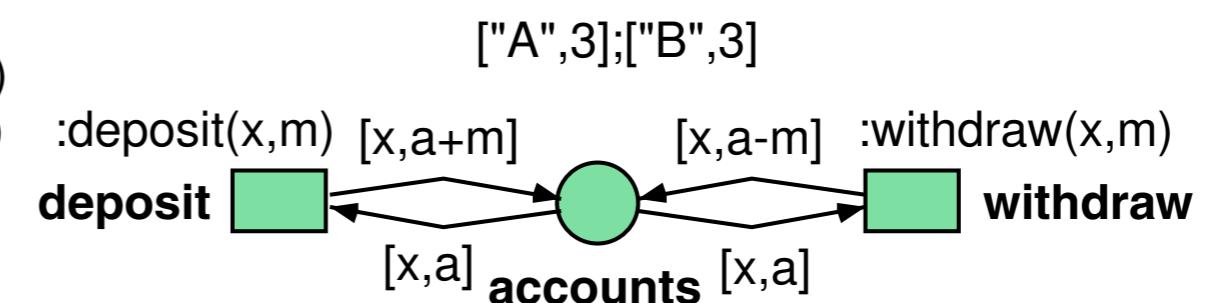


Abbildung 4.28: Wiederverwendung eines Kanals

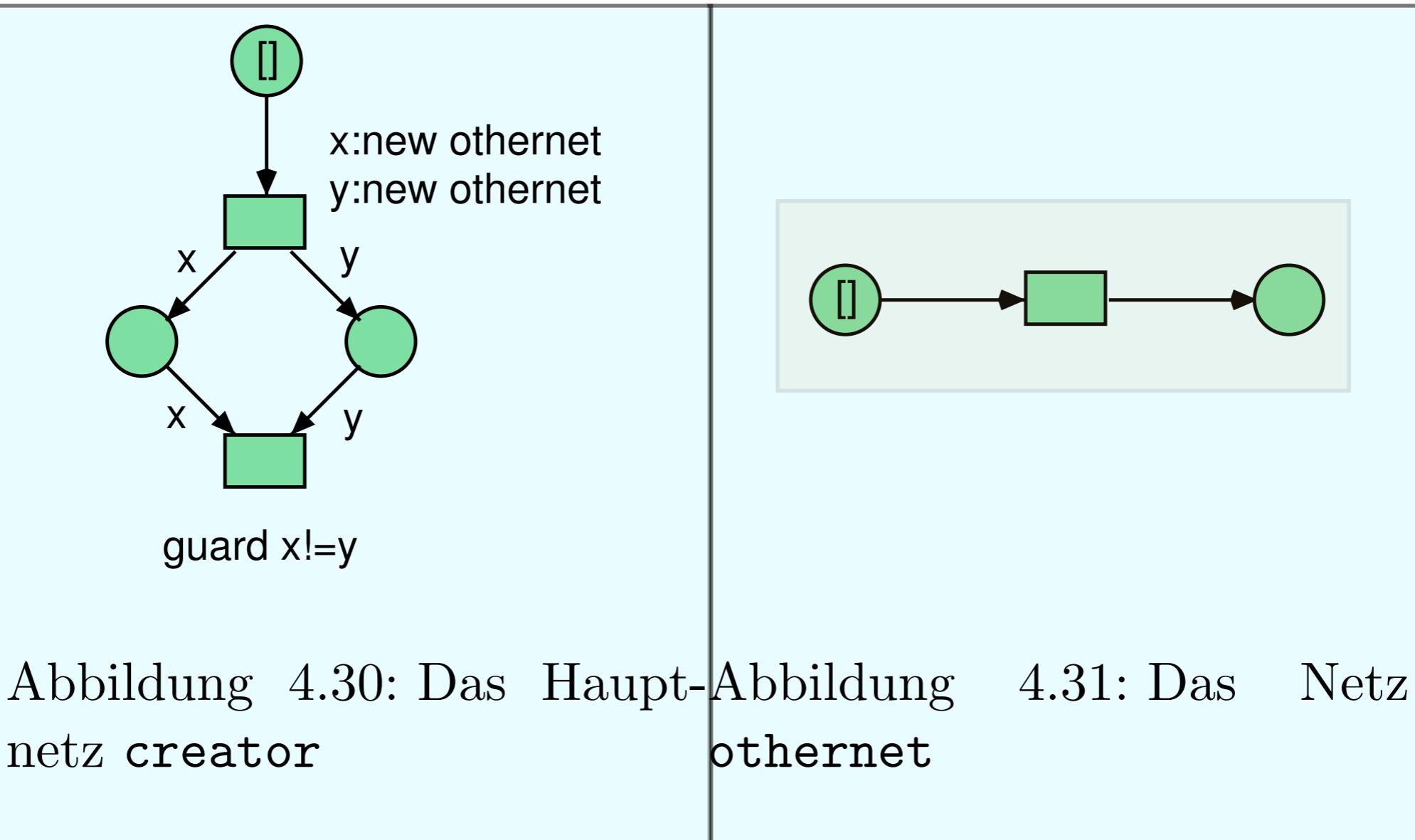


Abbildung



Buchführung mit Algebra

Netzinstanzen



Netzreferenzen

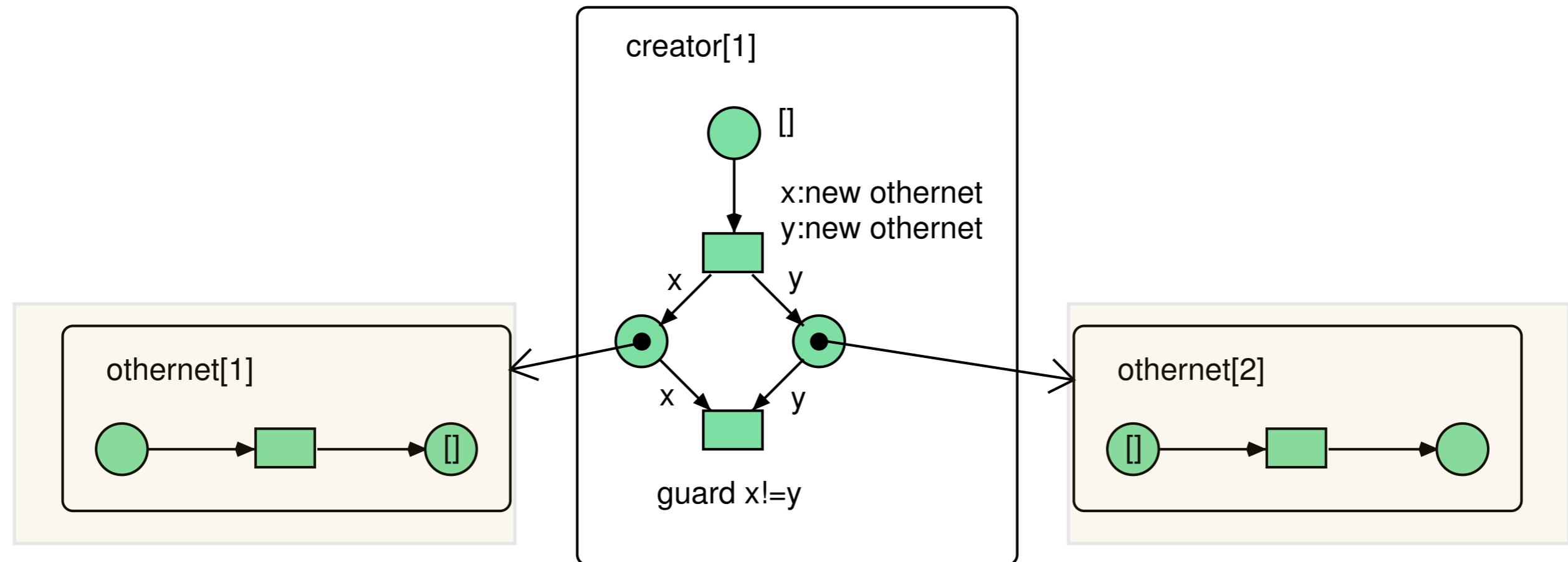
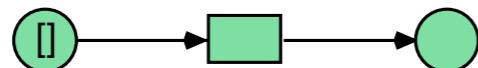
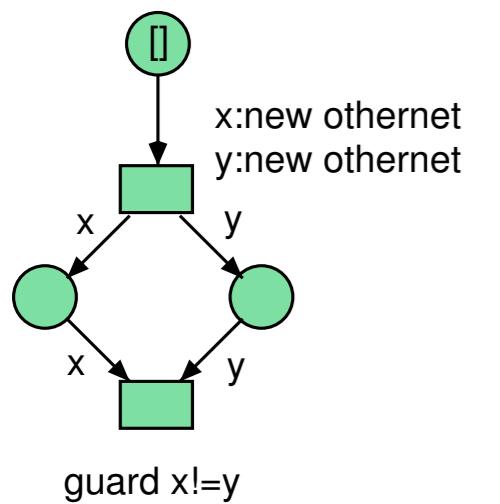


Abbildung 4.32: Netzexemplare mit Netzreferenzen

Variante

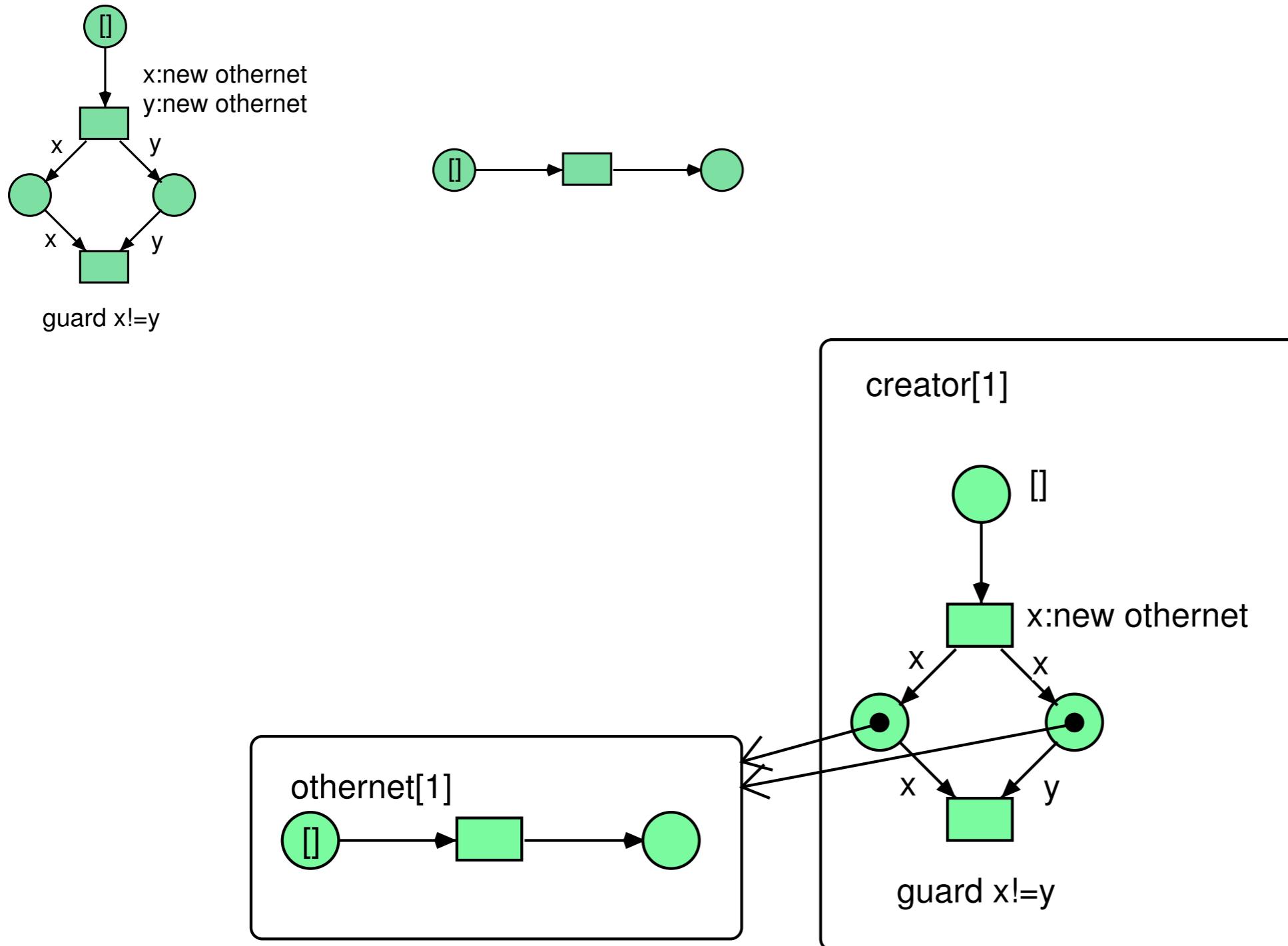
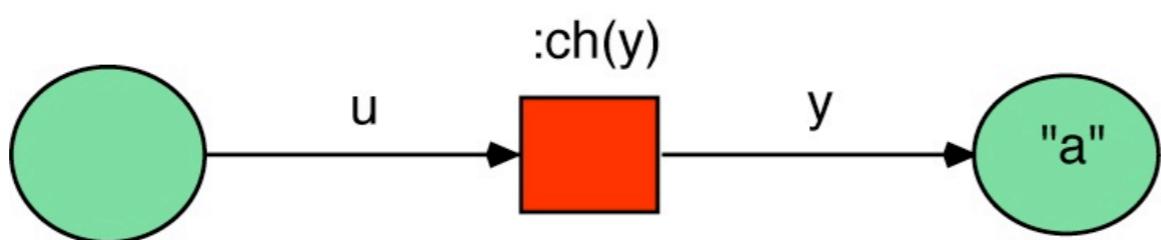
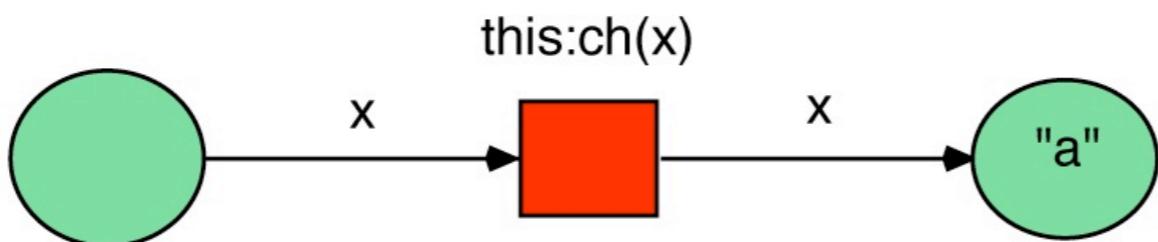
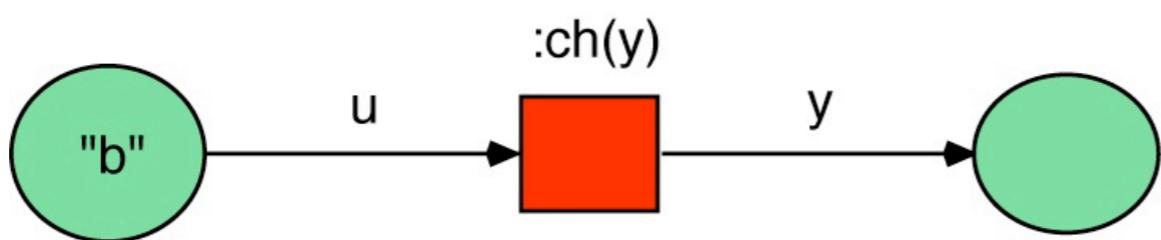
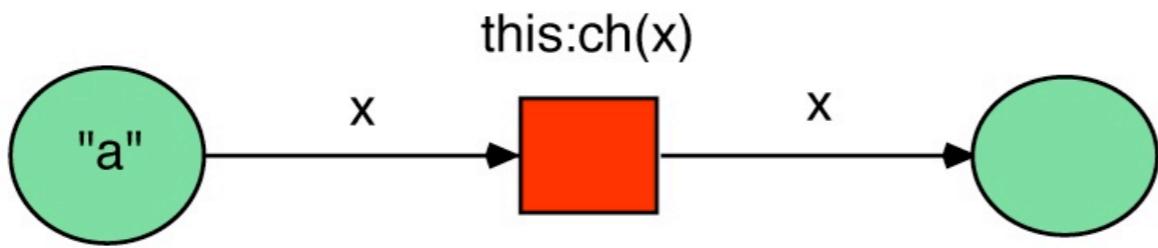
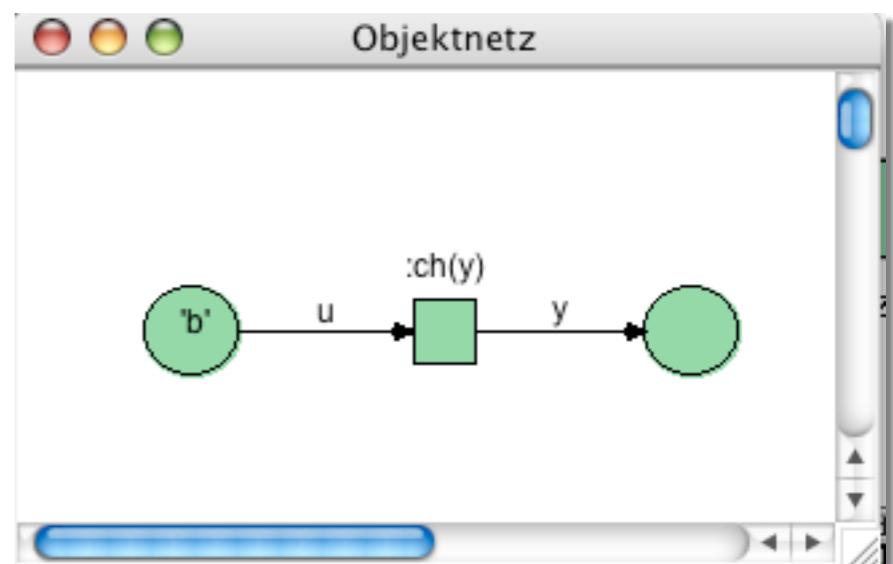
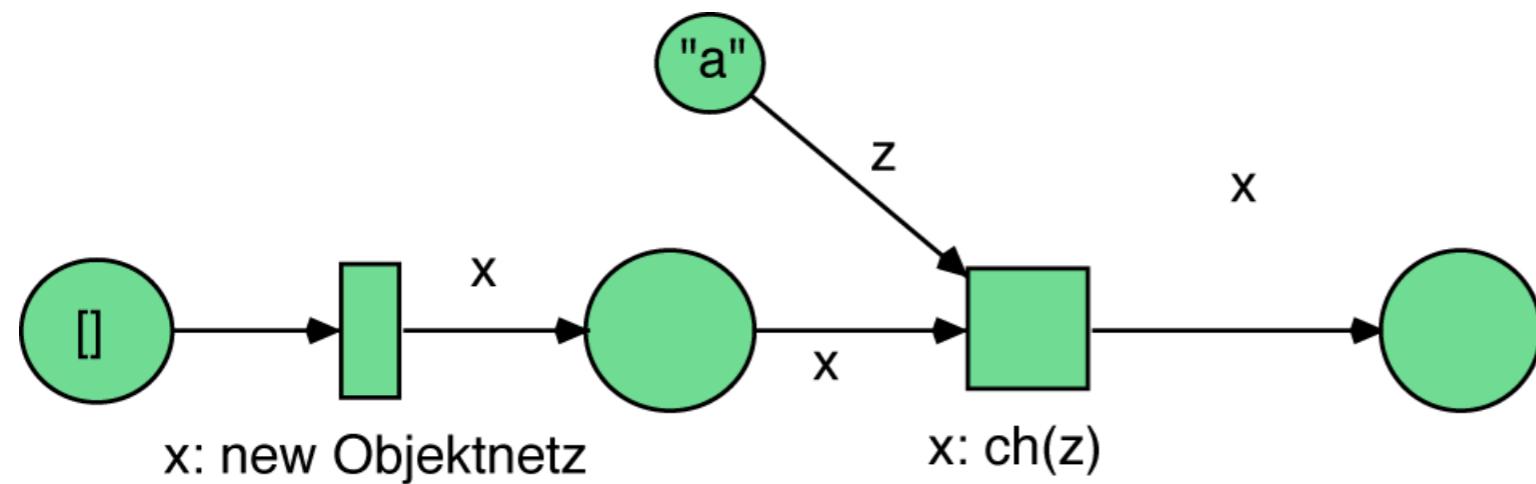


Abbildung 4.33: Netzexemplare mit Netzreferenzen: Variante

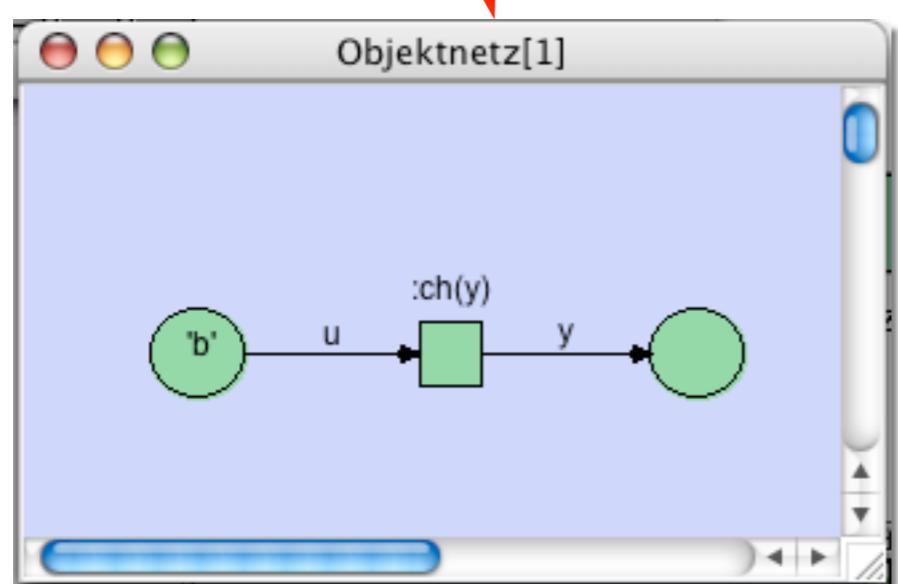
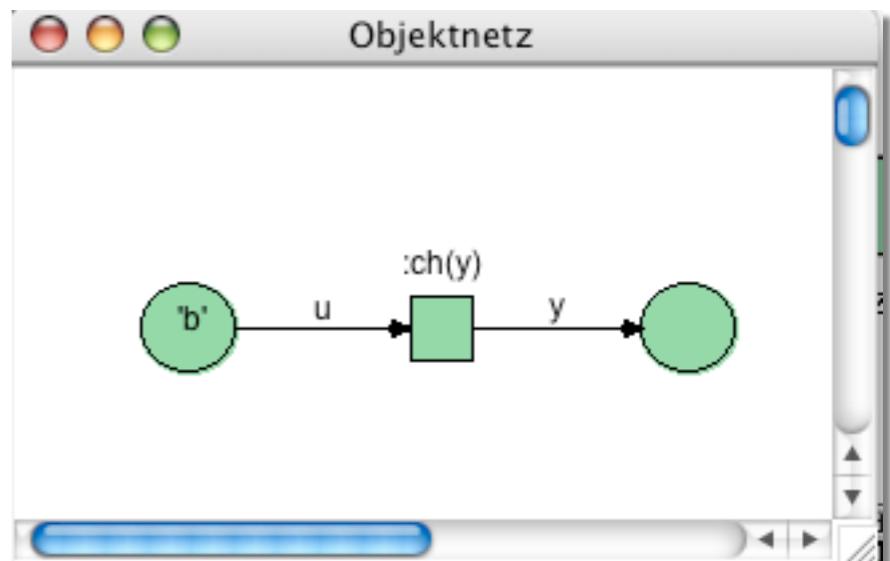
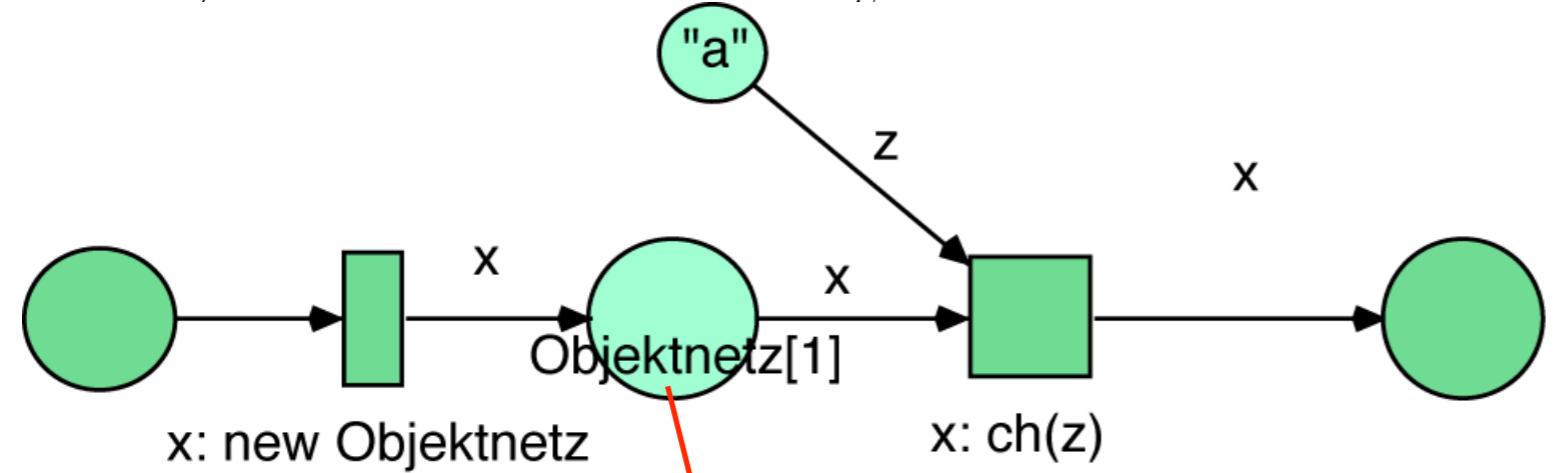
Synchrone Kanäle mit Parameterübergabe



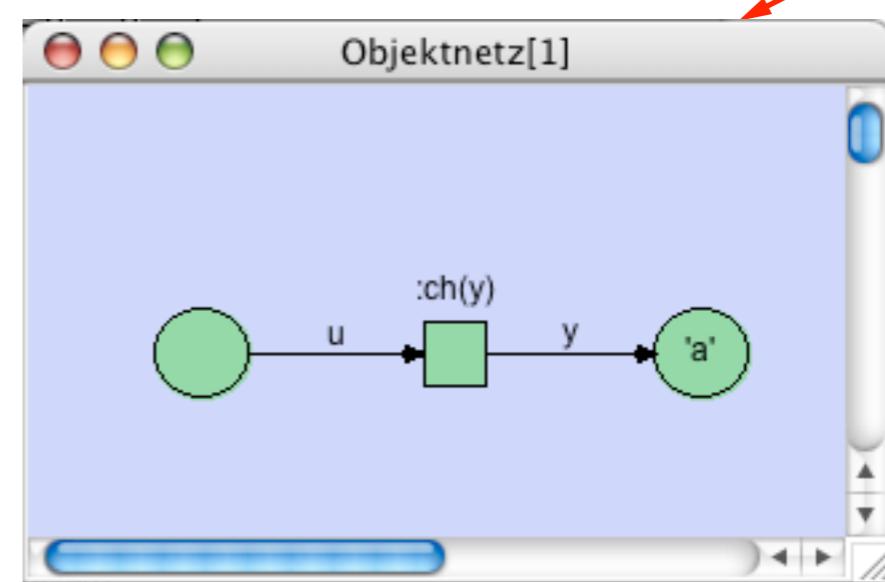
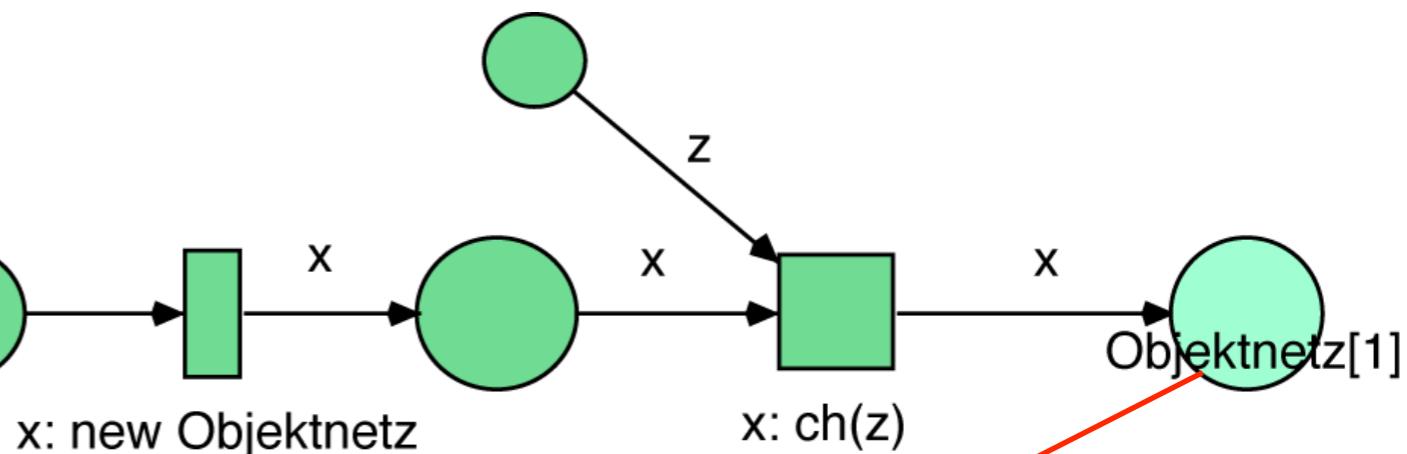
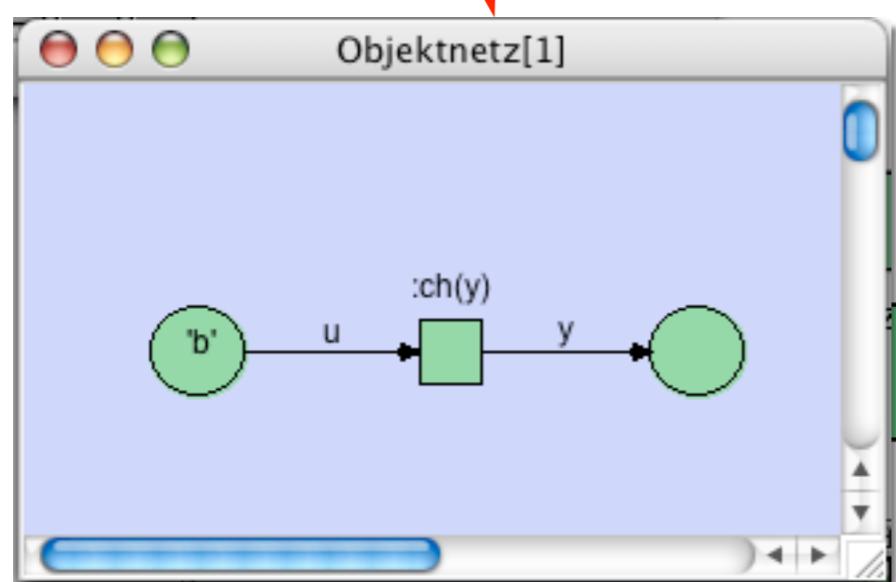
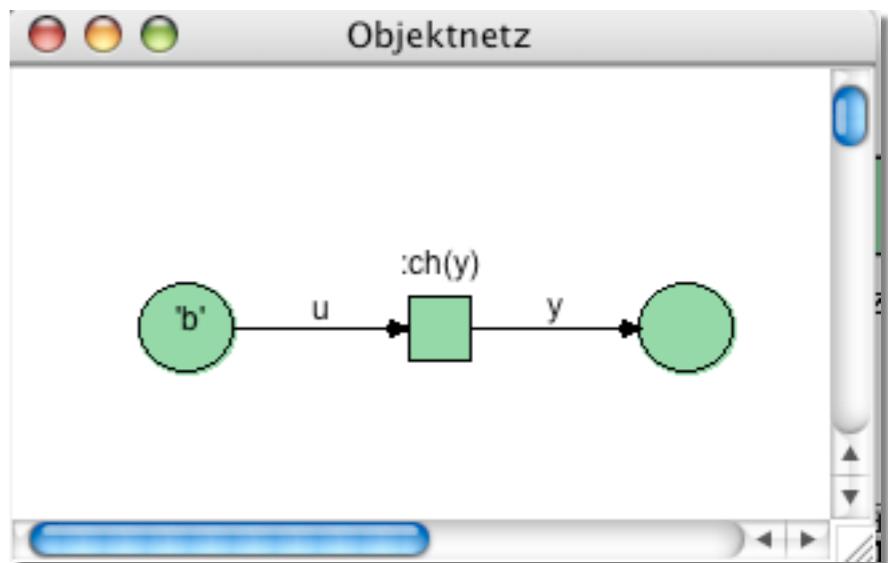
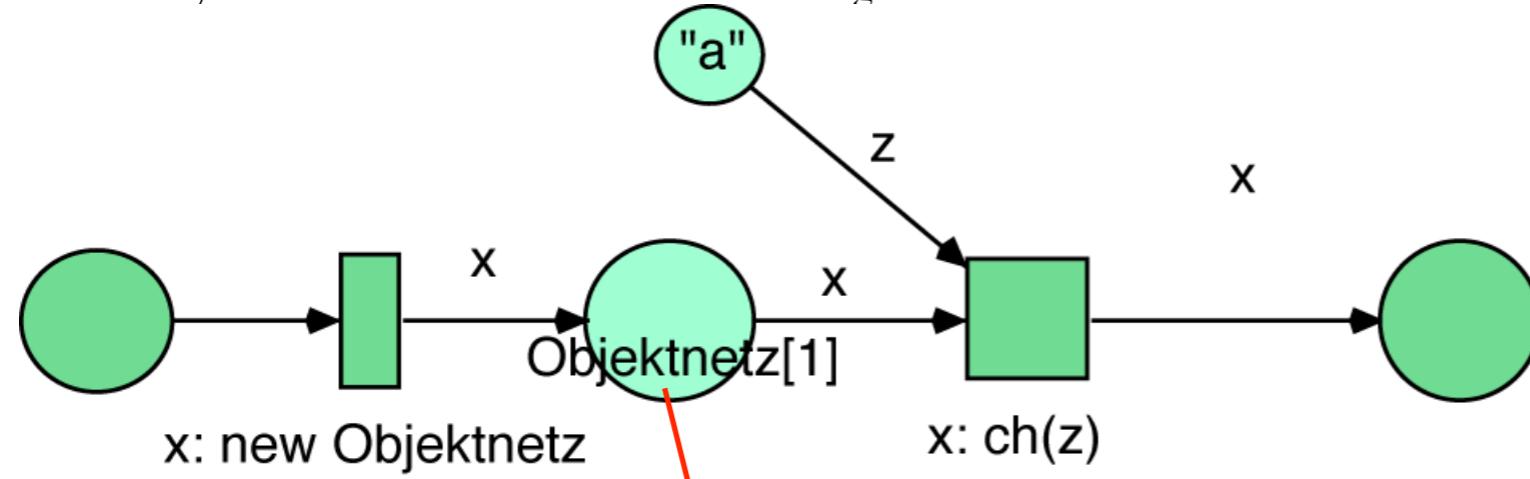
Reference Nets and Renew-Tool: synchronous channel and net referencing



Renew-Tool: synchronous channel and net referencing



Renew-Tool: synchronous channel and net referencing

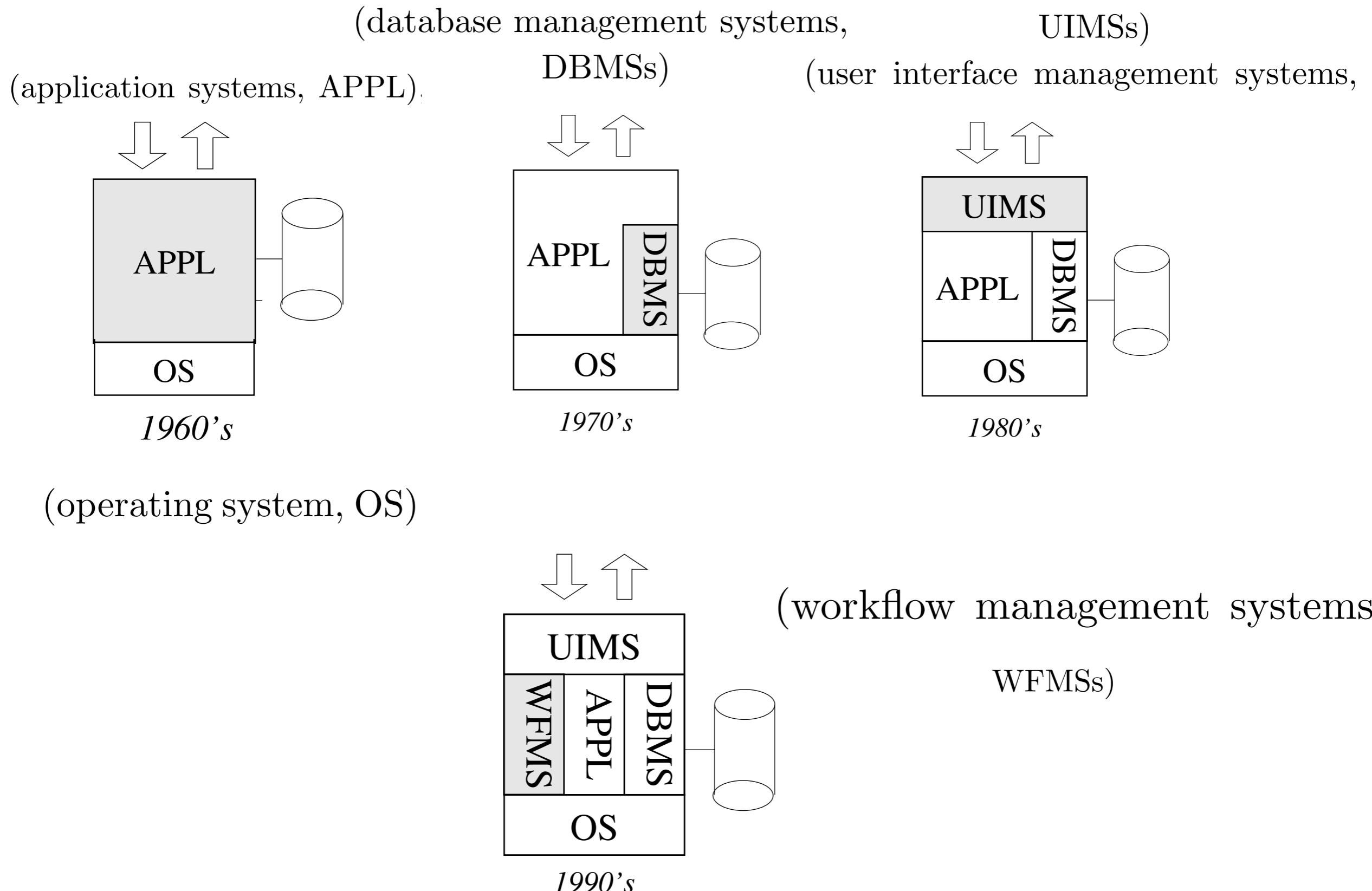


FGI 2

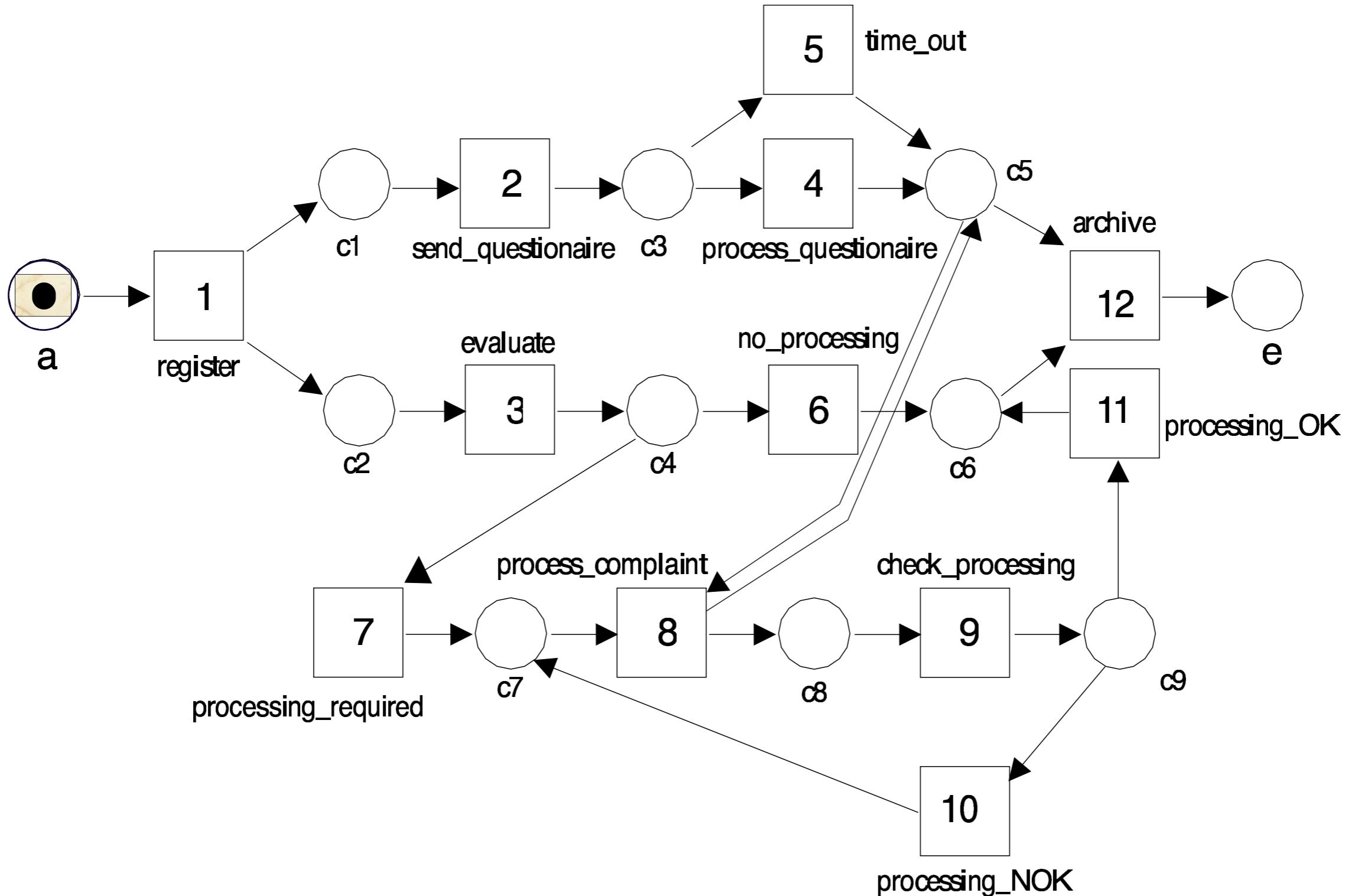
Daniel Moldt

Workflow

Workflow-Systeme

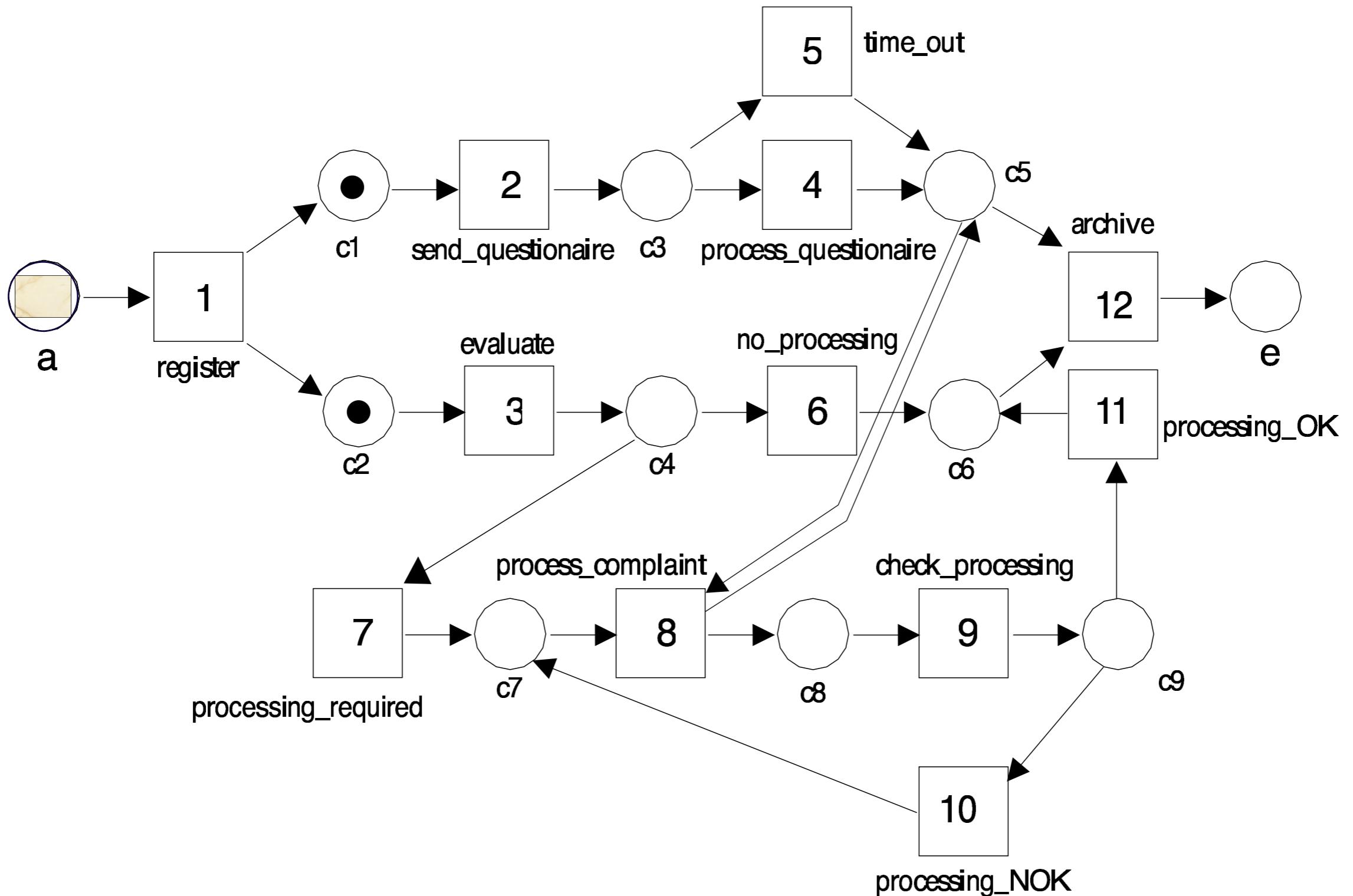


Beispiel



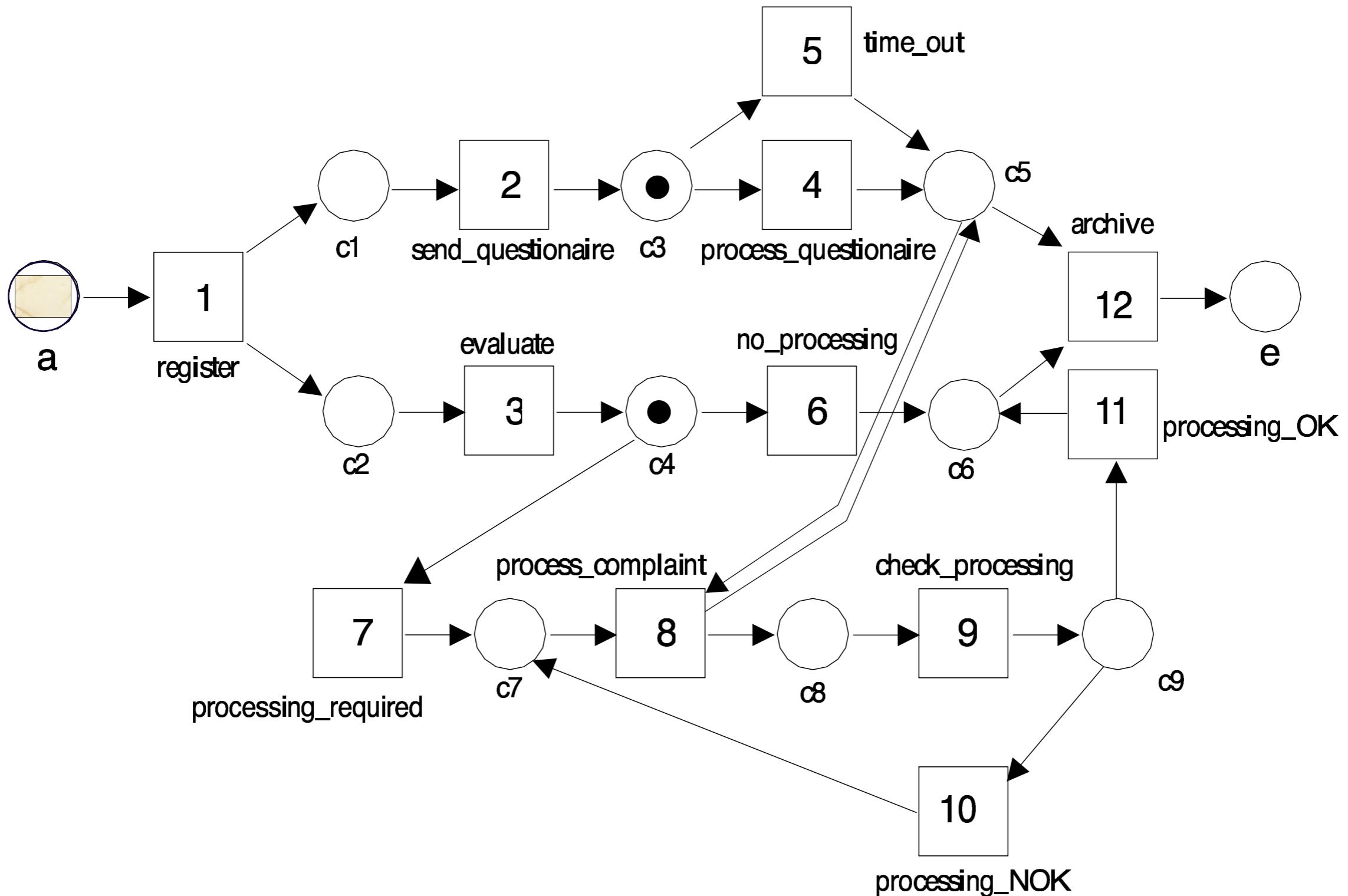
1. Aufnahme einer Beschwerde (register)

Beispiel



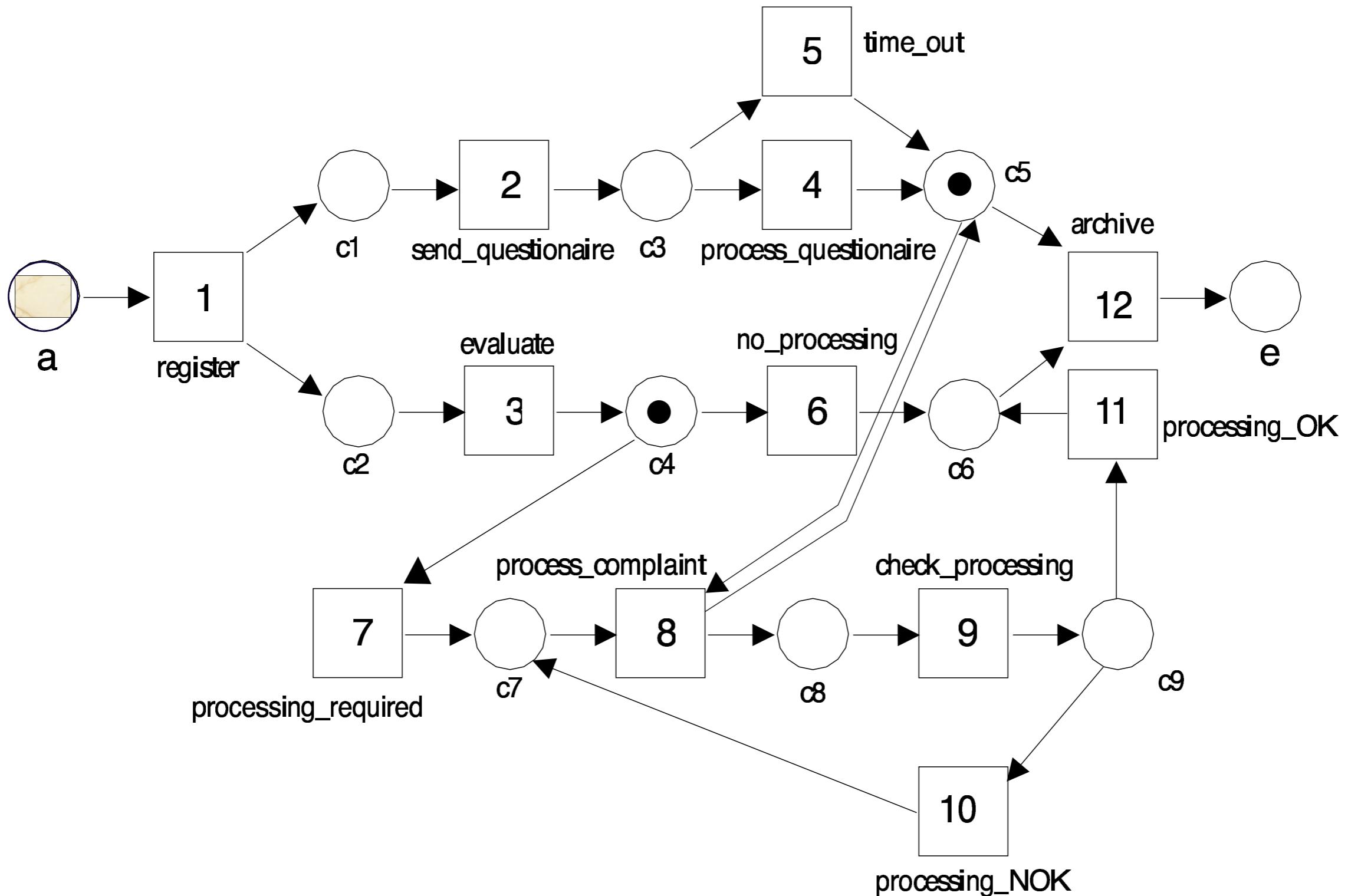
2. Fragebogen an Beschwerdeführer (send_questionnaire)
 3. Bewertung (evaluate) (nebenläufig zu 2.)

Beispiel



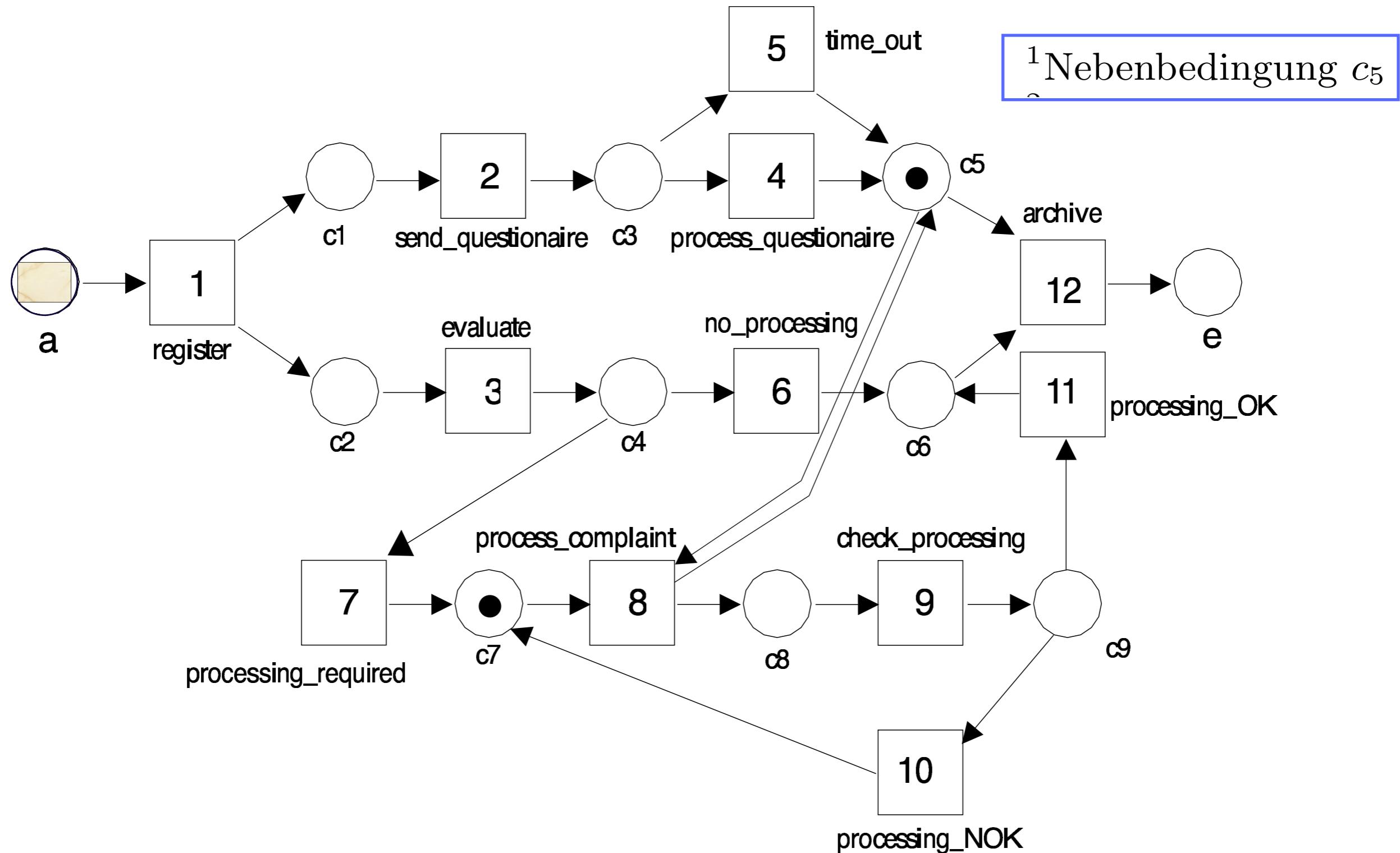
4. Fragebogenauswertung (process_questionnaire), falls Rücklauf innerhalb von 2 Wochen, sonst:
5. Nichtberücksichtigung des Fragebogens (time_out).

Beispiel



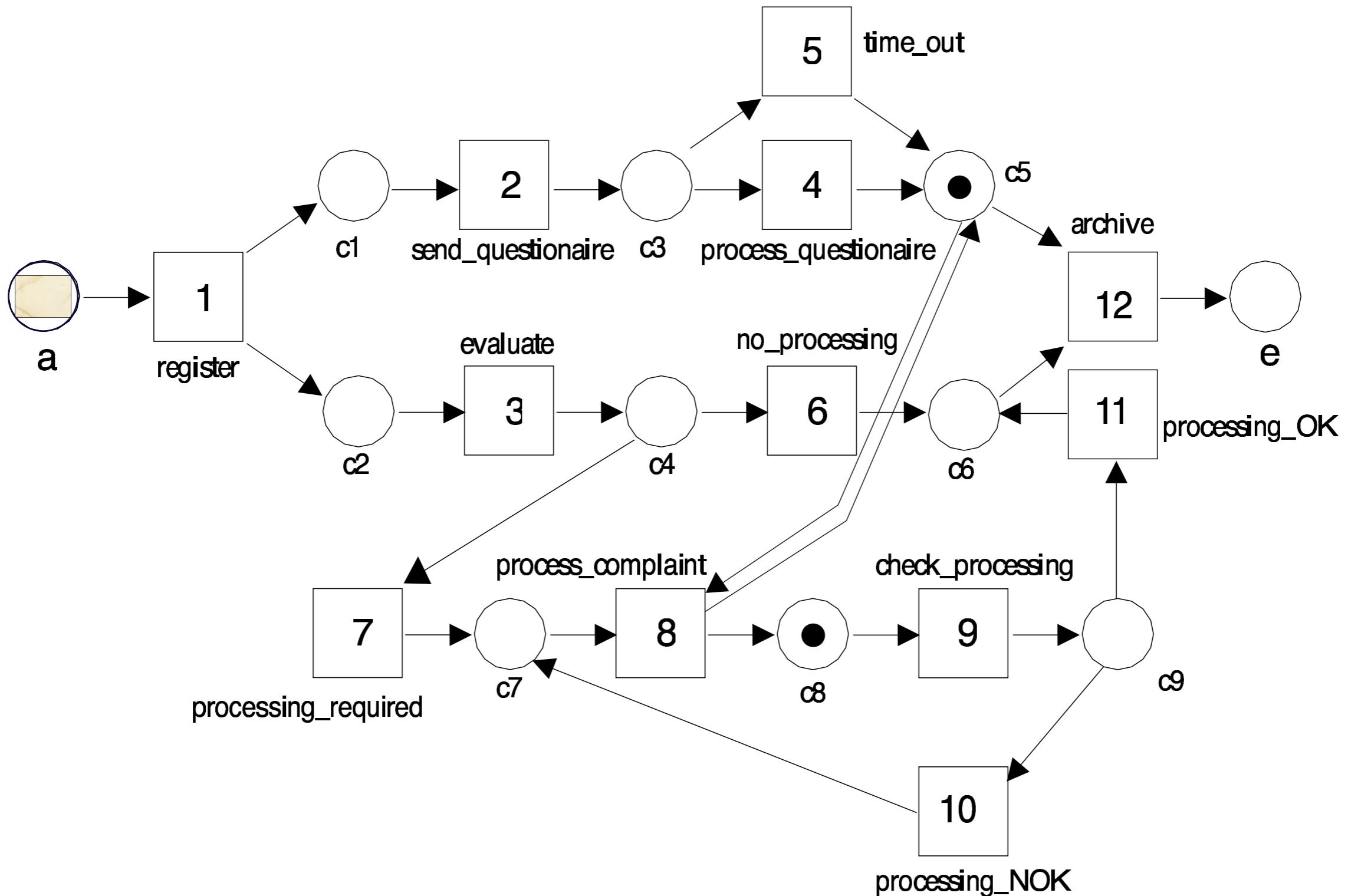
6. Je nach Ergebnis der Bewertung (3.) : Aussetzung der Bearbeitung (no_processing) oder
7. Beginn der eigentlichen Prüfung (processing_required)

Beispiel



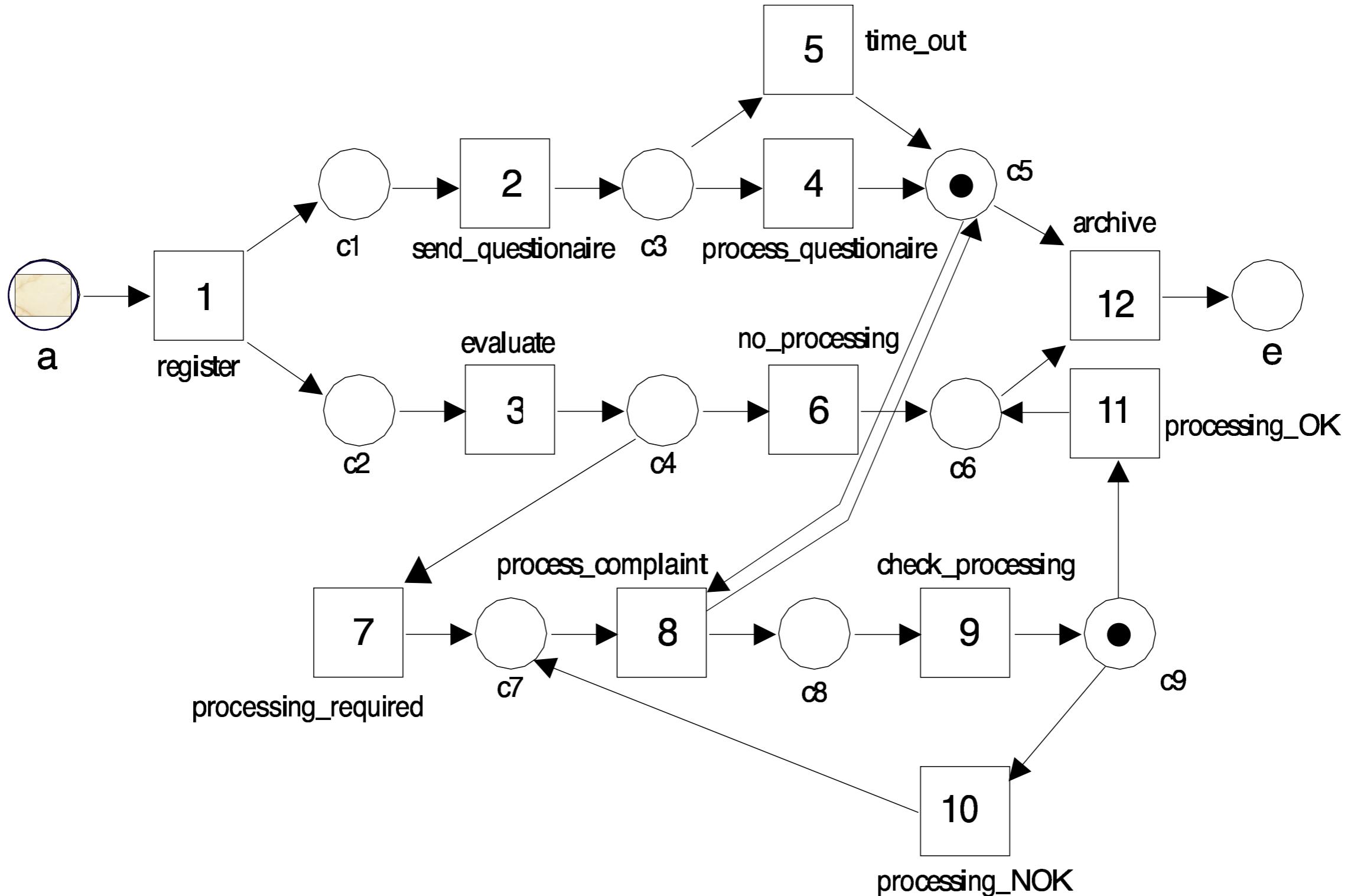
8. Bearbeitung der Beschwerde (process_complaint) unter Berücksichtigung des Fragebogens¹

Beispiel



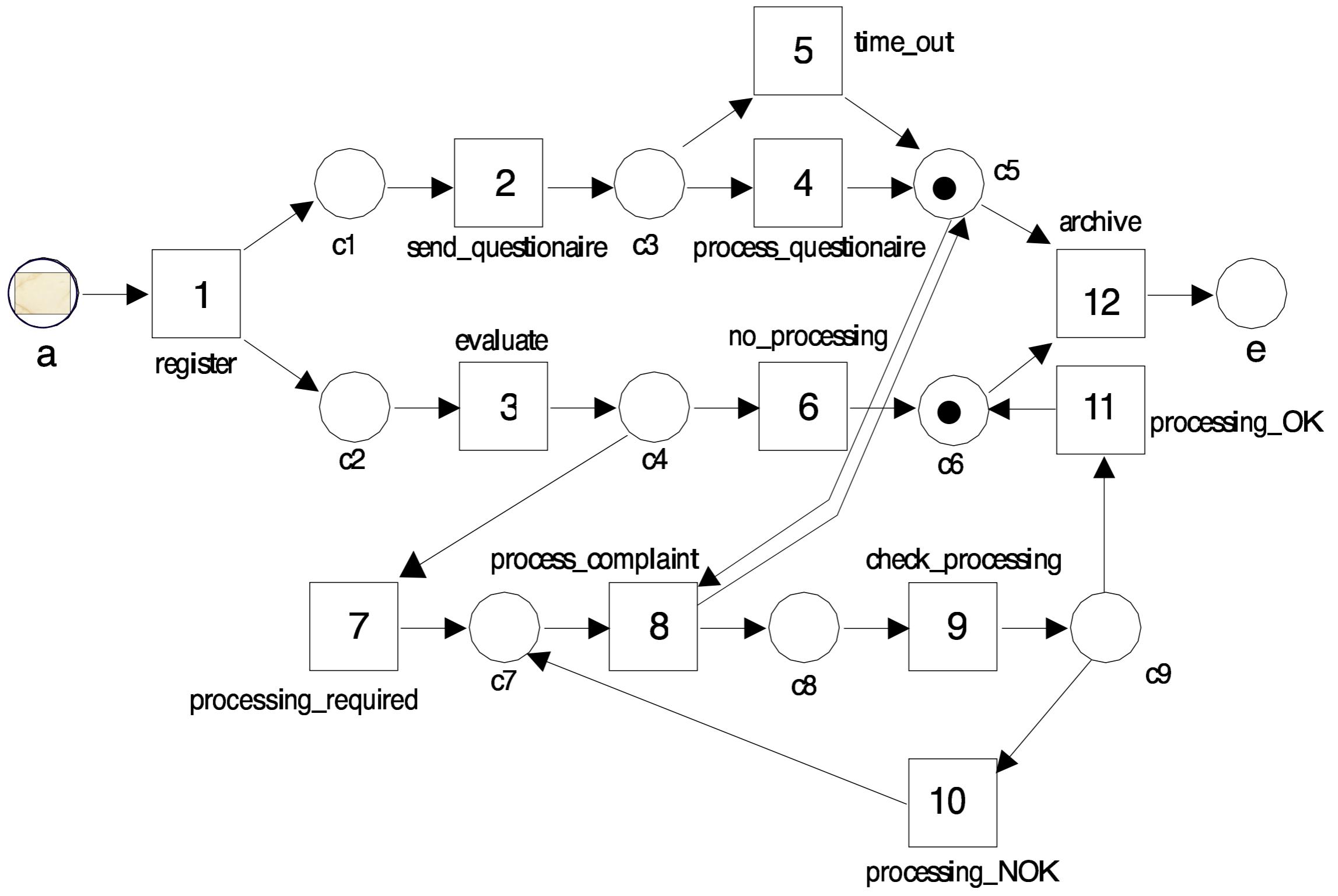
9. Bewertung der Bearbeitung (check_processing) mit dem Ergebnis

Beispiel

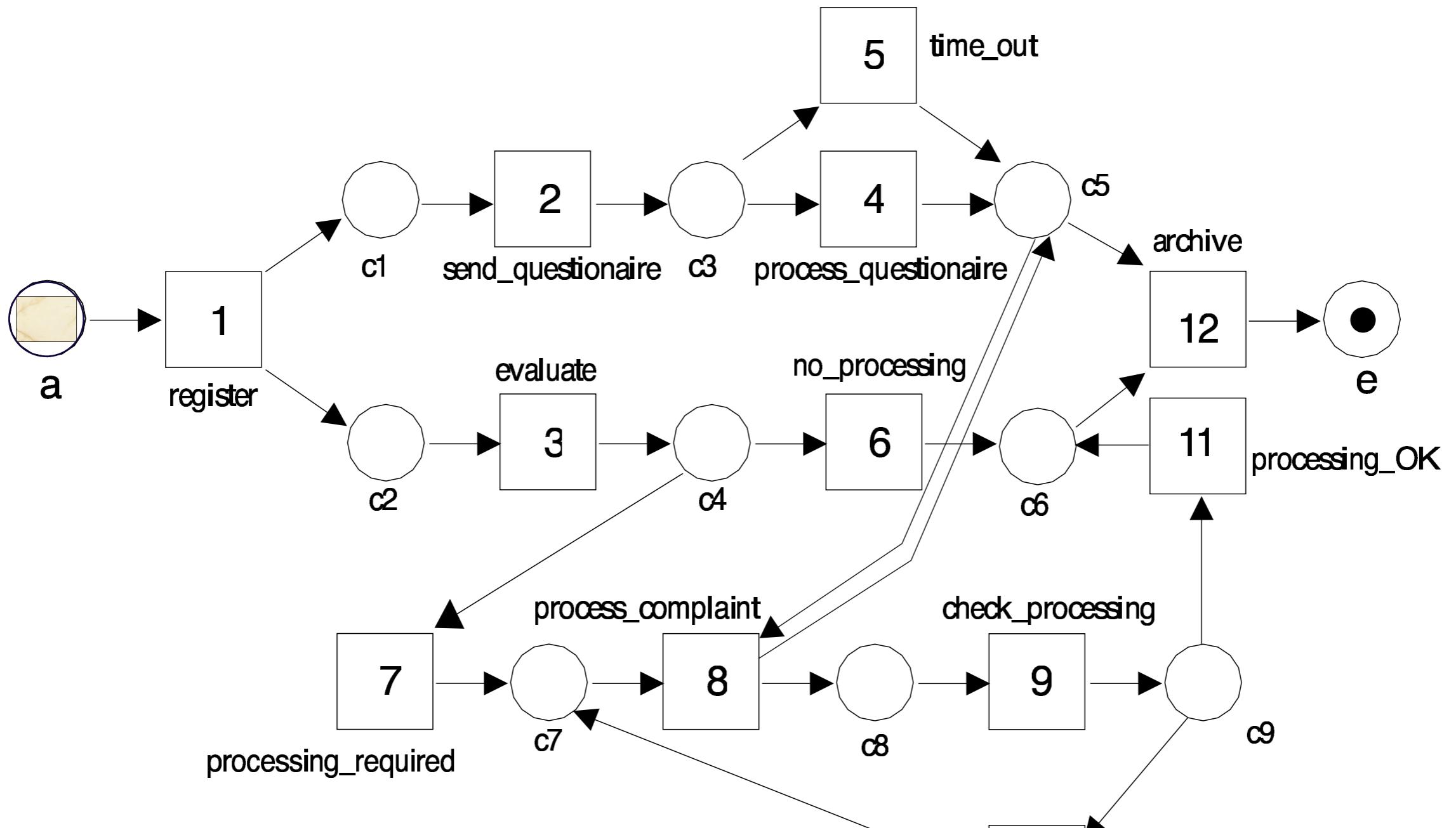


10. erneute Prüfung (processing_nok) oder
 11. Abschluss (processing_ok)

Beispiel



12. Ablage (archive)

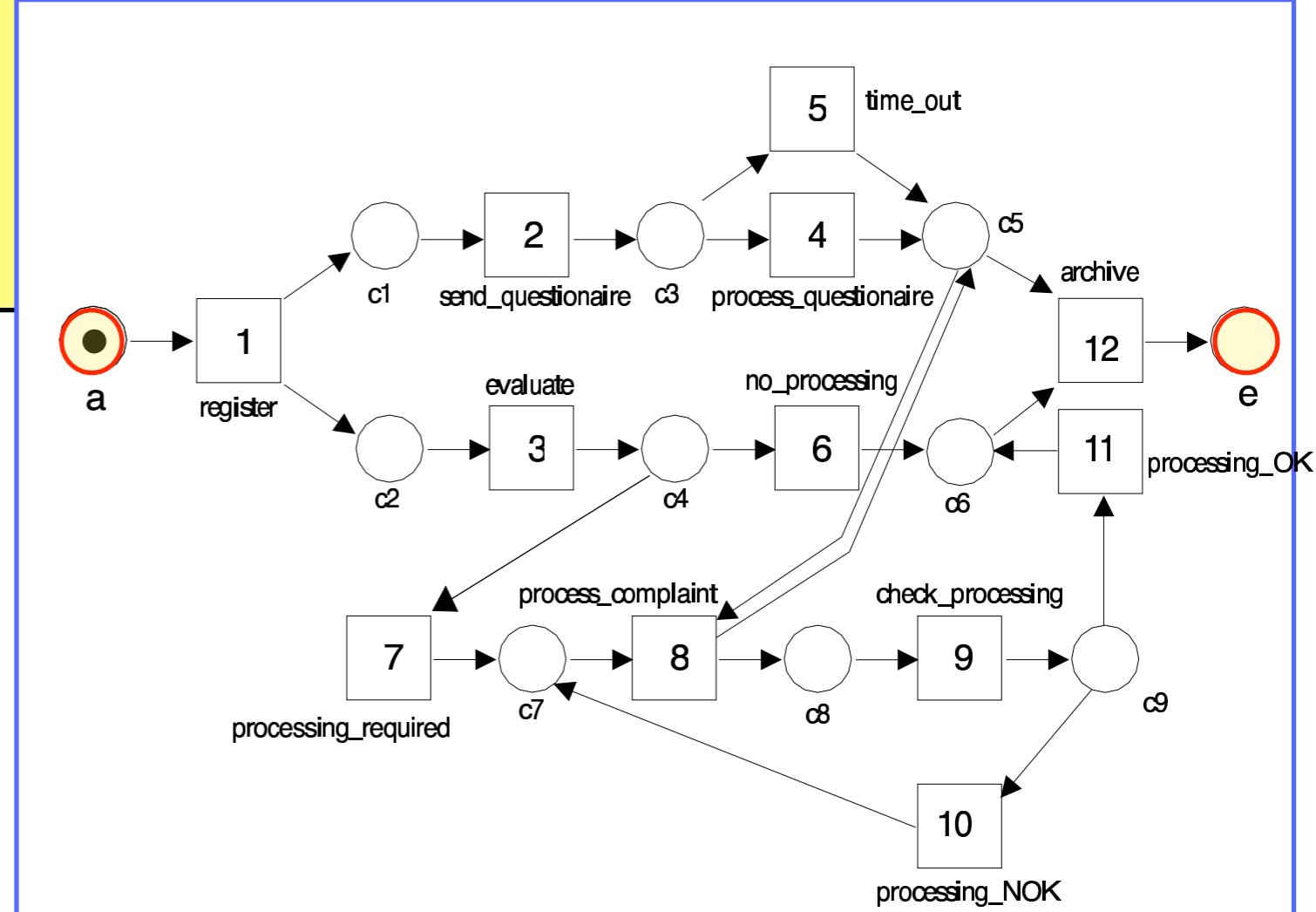


12. Ablage (archive)

Workflow-Netze

Definition 2.17 Ein P/T -Netz $\mathcal{N} = (P, T, F, \mathbf{m}_a)^2$ heißt Workflow-Netz (WF-Netz), falls

- a) es zwei besondere Plätze $\{a, e\} \subseteq P$ enthält mit $\bullet a = \emptyset$ („Start, Quelle, Anfangsplatz“) und $e^\bullet = \emptyset$ („Ende, Senke, Endplatz“),
- b) alle Plätze und Transitionen auf Pfaden zwischen a und e liegen und
- c) Anfangsmarkierung \mathbf{m}_a
Endmarkierung \mathbf{m}_e



Mehrere Fälle

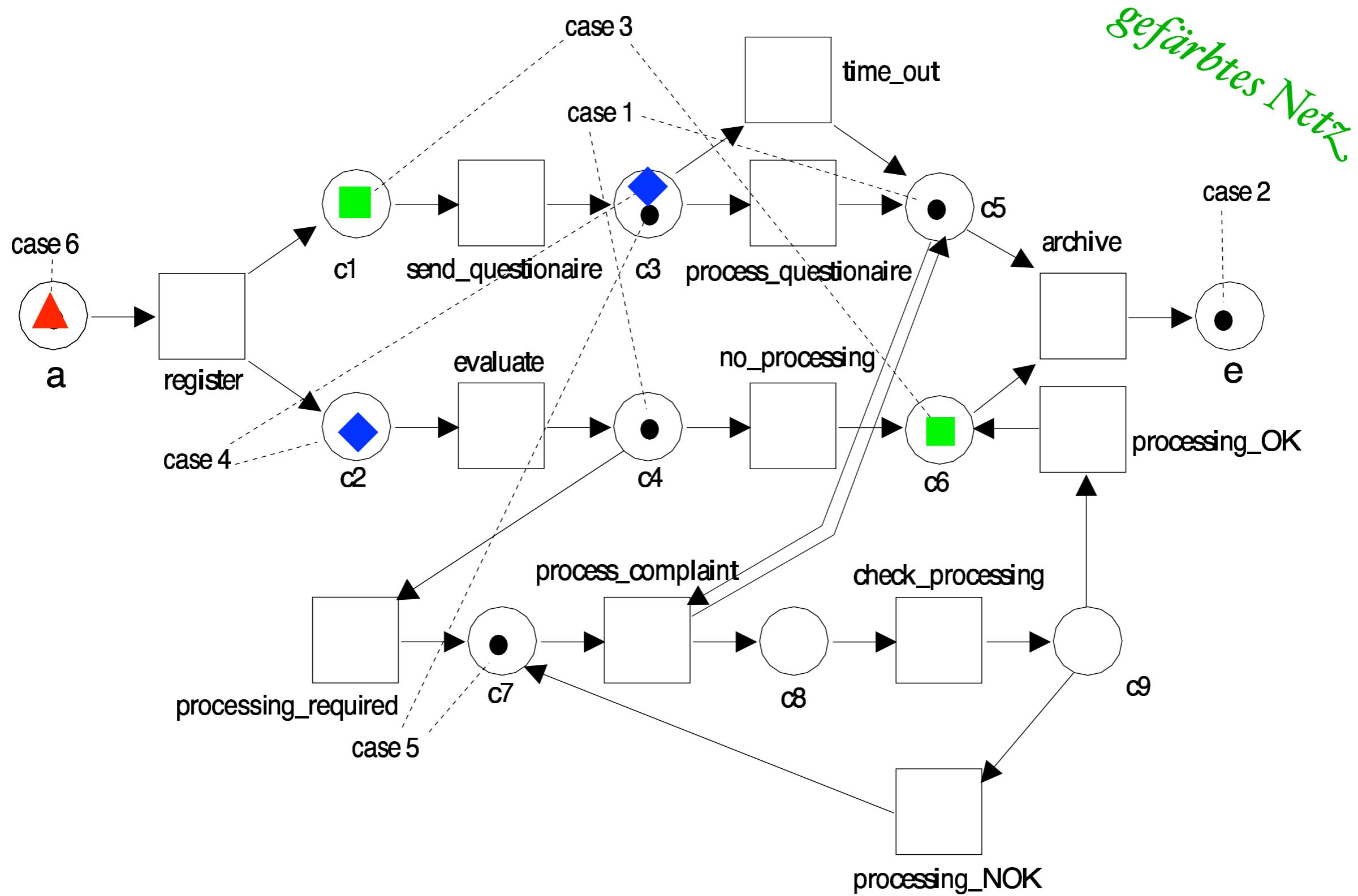


Abbildung 2.26: Gefärbtes Netz zur Darstellung mehrerer Fälle in einem Netz

Typische Strukturen („patterns³“)

<http://www.workflowpatterns.com/patterns>

Pattern

[Patterns](#)
[Evaluations](#)
[Vendors](#)
[About...](#)
[Impact](#)
[YAWL](#)
[Links](#)
[Documentation](#)
[Contacts](#)
[Site Map](#)

Pattern 2 (Parallel Split)

[FLASH animation of Parallel Split pattern](#)

Description

The divergence of a branch into two or more parallel branches each of which execute concurrently.

Synonyms

AND-split, parallel routing, parallel split, fork.

Examples

After completion of the *capture enrolment* task, run the *create student profile* and *issue enrolment confirmation* tasks simultaneously.

When an *intrusion alarm* is received, trigger the *despatch patrol* task and the *inform police* task immediately.

Once the customer has paid for the goods, pack them and issue a receipt.

Motivation

The *Parallel Split* pattern allows a single thread of execution to be split into two or more branches which can execute tasks concurrently. These branches may or may not be re-synchronized.

Overview

Figure 2 illustrates the implementation of the *Parallel Split*. After task A has completed, two distinct threads of execution are initiated and tasks B and C can proceed concurrently.

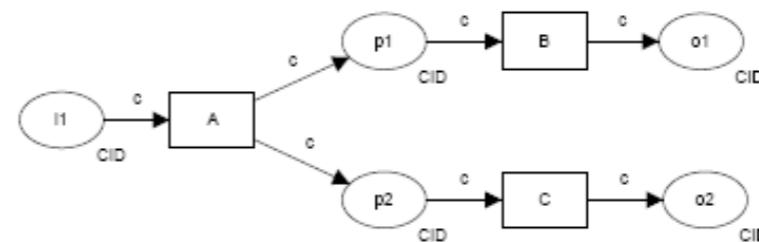


Figure 2: Parallel split pattern

Context

There are no specific context conditions for this pattern.

Implementation

The *Parallel Split* pattern is implemented by all of the offerings examined. It may be depicted either *explicitly* or *implicitly* in process models. Where it is represented explicitly, a *Split* with one incoming edge and two or more outgoing edges. Where it is represented implicitly, this can be done in one of two ways: either (1) the edge representing control-branches or (2) the task after which the *Parallel Split* occurs has multiple outgoing edges which do not have any conditions associated with them or where it does these conditions alwa

Pattern

Evaluation Criteria

Full support for this pattern is demonstrated by the provision of a construct (either implicit or explicit) that allows the thread of control at a given point in a process to be split into multiple parallel paths.

Product Evaluation

To achieve a + rating (direct support) or a +/- rating (partial support) the product should satisfy the corresponding evaluation criterion of the pattern. Otherwise a - rating is awarded.

Product/Language	Version	Score	Motivation
Staffware	10	+	Supported through a step construct that has multiple outgoing arcs.
Websphere MQ	3.4	+	Supported through multiple outgoing arcs from an activity.
FLOWer	3.51	+	Nodes in static, dynamic and sequential subplans have an AND-split semantics.
COSA	5.1	+	Supported by multiple outgoing arcs from an activity. None of the arcs have transitions.
iPlanet	3.0	+	Supported by multiple outgoing routers from an activity
SAP Workflow	4.6c	+	Directly supported. SAP allows for structured parallel processes, using the fork construct. Since there has to be a one-to-one correspondence between splits and joins, some of them are structured.
FileNet	3.5	+	Directly supported by a step where all outgoing routes are unconditional.
BPEL	1.1	+	Supported by <flow> construct.
Websphere Integration Developer	6.0	+	Supported by the <flow> activity.
Oracle BPEL	10.1.2	+	Supported by the <flow> construct.
BPMN	1.0	+	Supported by AND-split gateway
XPDL	2.0	+	Supported by the AND-split construct
UML ADs	2.0	+	Supported by the ForkNode construct. It may also be represented implicitly by join actions or activities.
EPC (implemented by ARIS toolset 6.2)		+	Supported by the AND-split connector.
jBPM	3.1.4	+	jBPM supports parallel split through: i) the <i>Fork Node</i> construct, which splits a tread to multiple treads running in parallel. ii) by defining the tasks to be run in parallel within the same <i>Task Node</i> .
OpenWFE	1.7.3	+	OpenWFE implements parallel split through the construct <concurrence>
Enhydra Shark	2	+	Enhydra Shark supports parallel split through i) an <i>activity</i> node with a <i>transition restriction</i> <Split Type = "AND"> specifying the parallel split. ii) a <i>routing activity</i> node with a <i>transition restriction</i> <Split Type = "AND"> specifying the parallel split. contrast to an activity, a routing activity does not implement any actions.

Sequenz, nebenläufige Bearbeitung

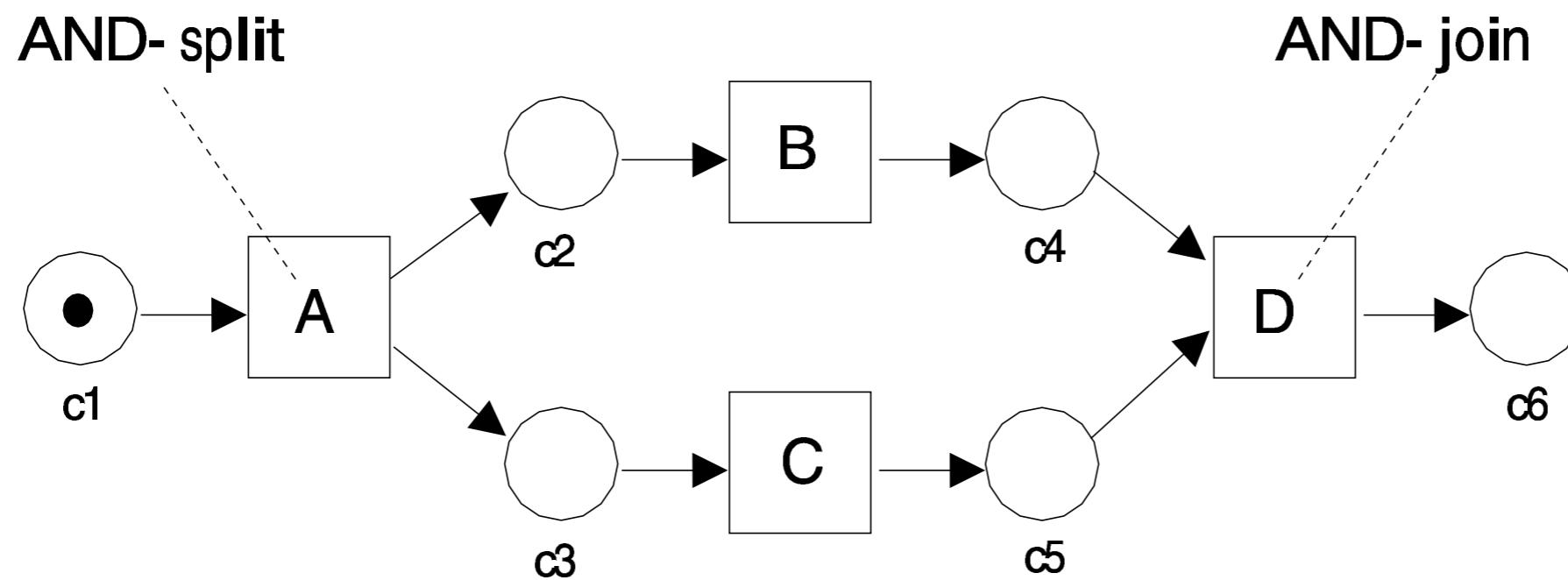
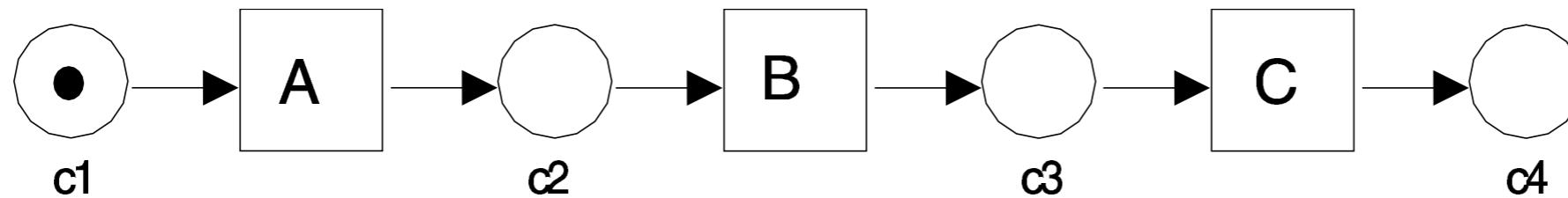


Abbildung 2.27: Sequentielle und nebenläufige Bearbeitung

Alternative und Iteration

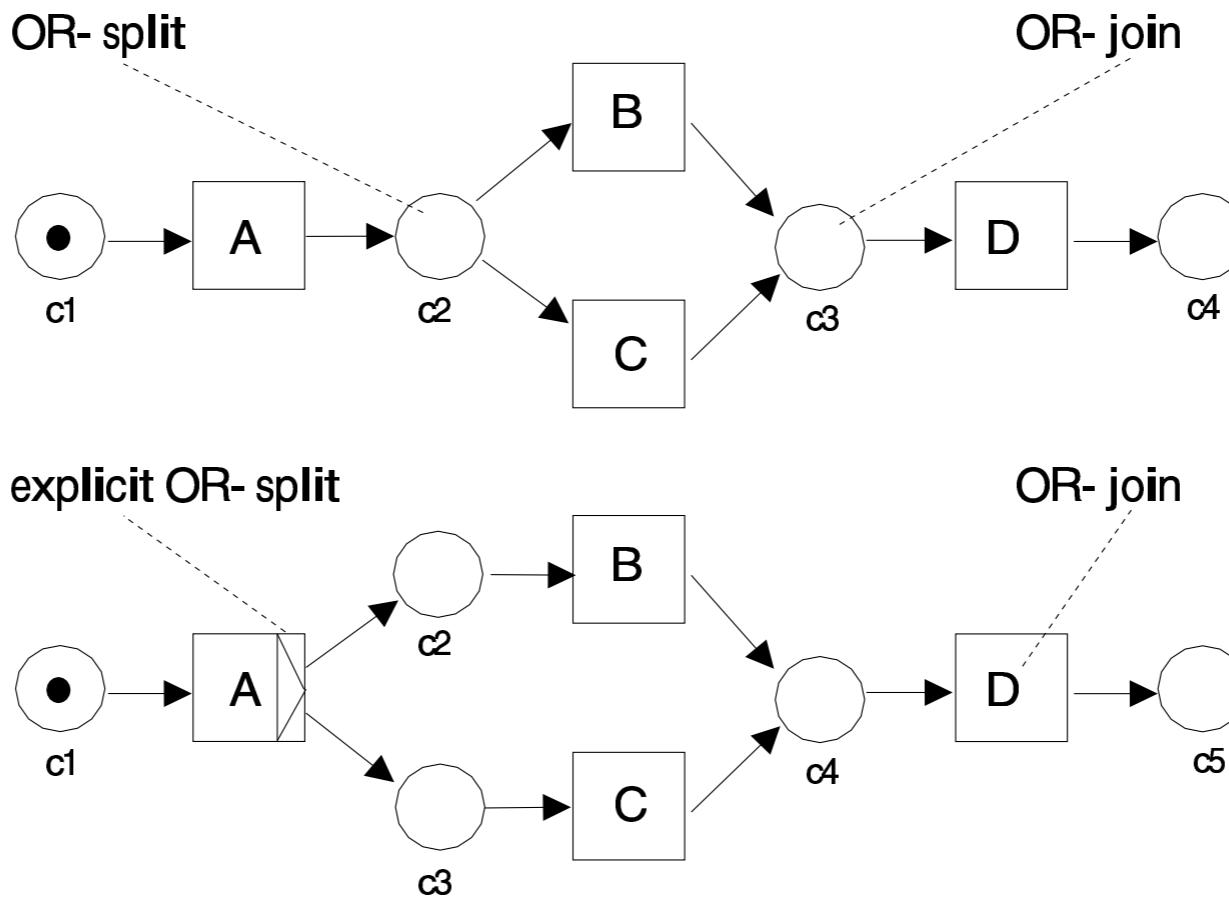


Abbildung 2.28: Alternative Bearbeitung mit und ohne expliziten Testbedingungen

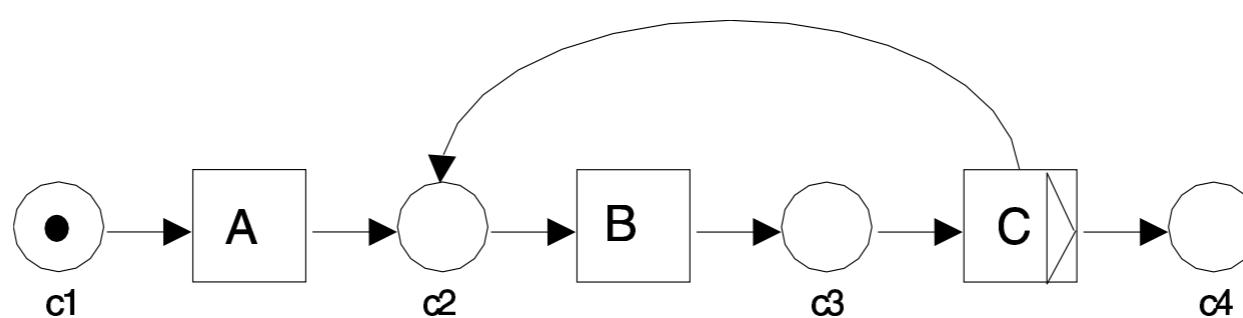


Abbildung 2.29: Iteration

Verzweigung

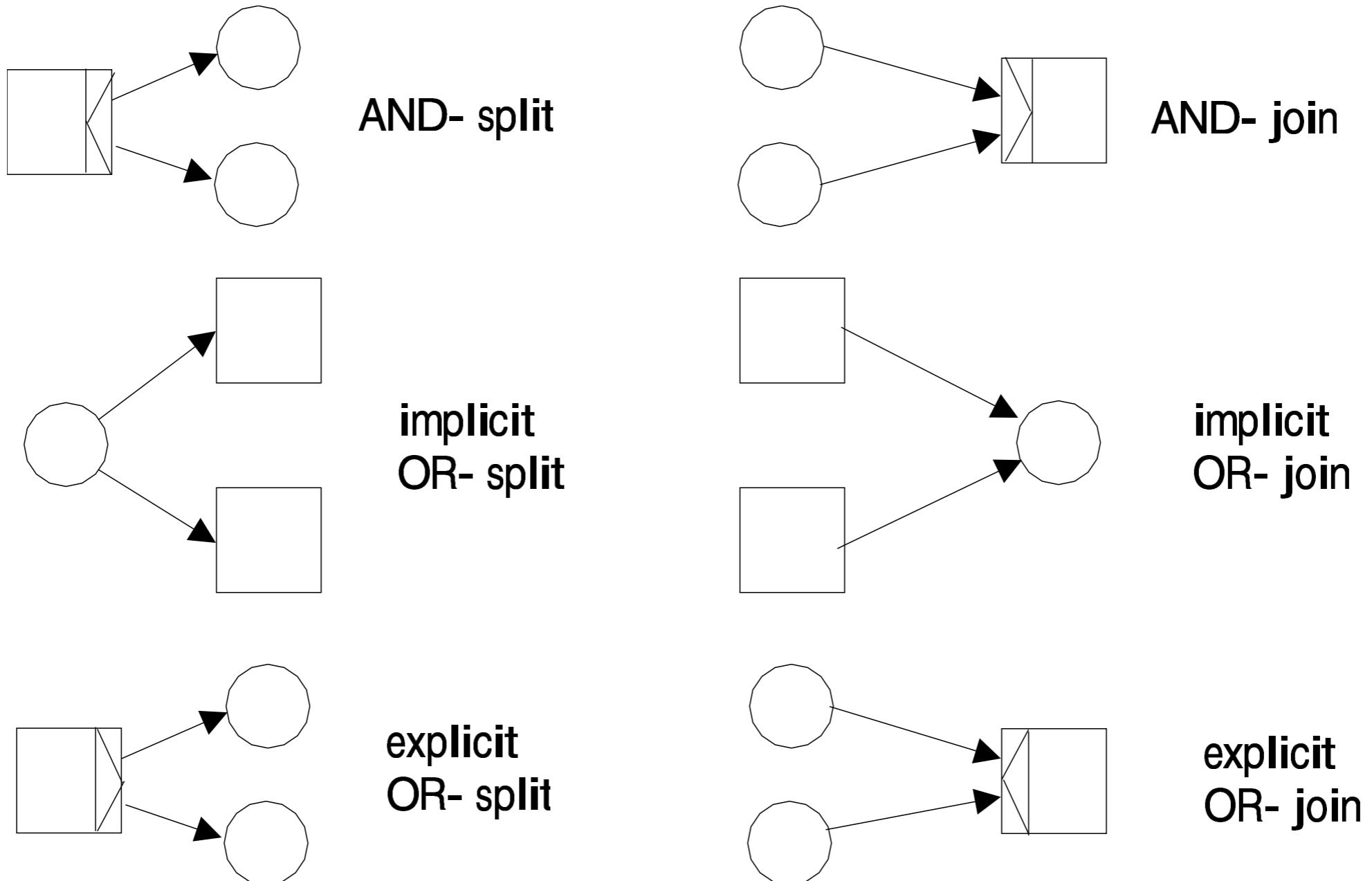


Abbildung 2.30: Graphische Symbole für Verzweigung

Trigger

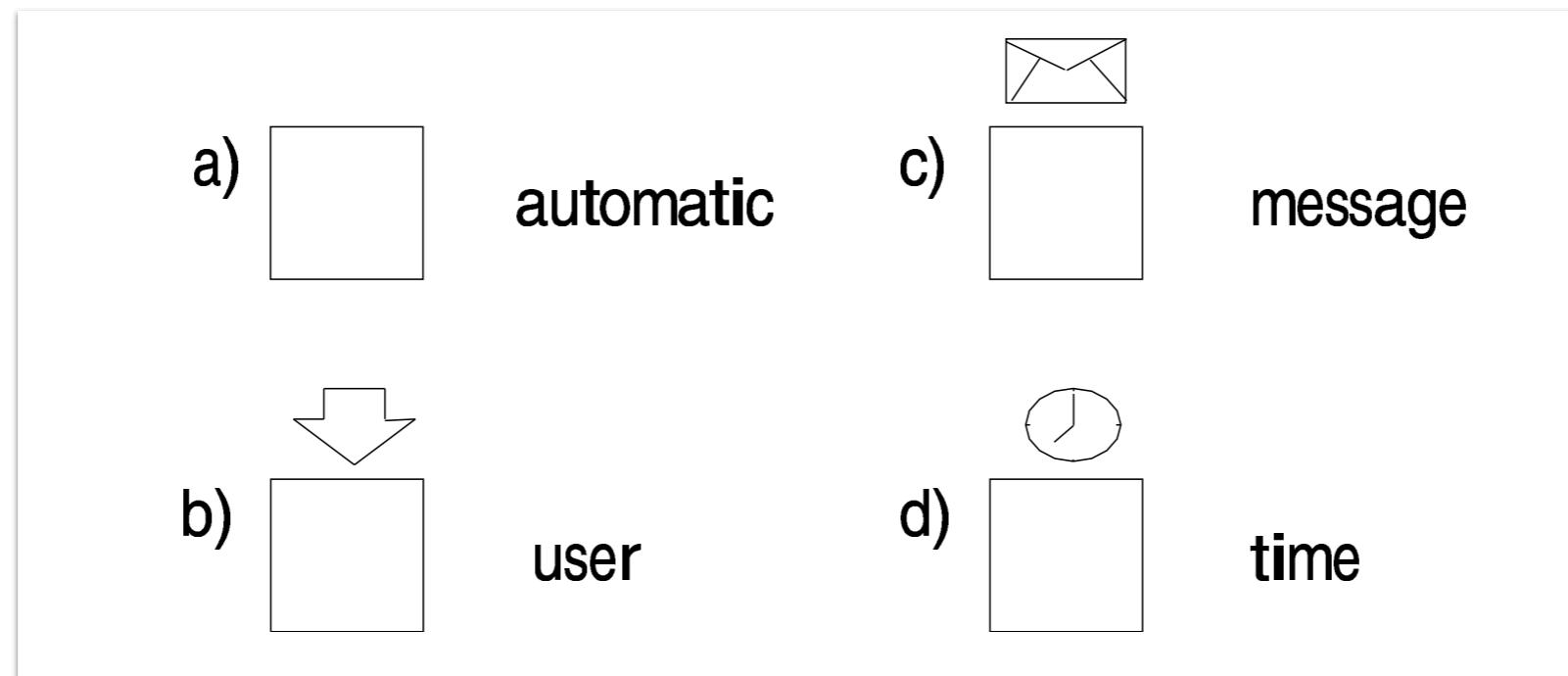


Abbildung 2.31: Trigger eines Workflowsystems

- a) Automatisch (keine externe Eingabe notwendig).
- b) Benutzer (user) : ein Bearbeiter oder Benutzer oder eine Funktionseinheit nimmt einen Auftrag an und bearbeitet ihn.
- c) Nachricht (message) : eine Nachricht von außen wird benötigt (Brief, Anruf, e-mail, Fax).
- d) Zeit (time) : es besteht eine Zeitbedingung für die Bearbeitung.

Vorzüge der Modellierung

Die Erfahrung aus der Praxis zeigt, dass

- Workflow-Prozesse oft nicht richtig verstanden werden (Mehrdeutigkeit, Widersprüche, Verklemmungen),
- allein schon die (versuchsweise) Modellierung durch Petrinetze Mängel aufdeckt und
- bei fertiggestellten Petrinetz-Modellen von Workflow-Systemen Mängel durch strukturelle Untersuchungen aufgedeckt oder durch Werkzeuge (automatisch) gefunden werden.

Beispiel: Problematischer Workflow

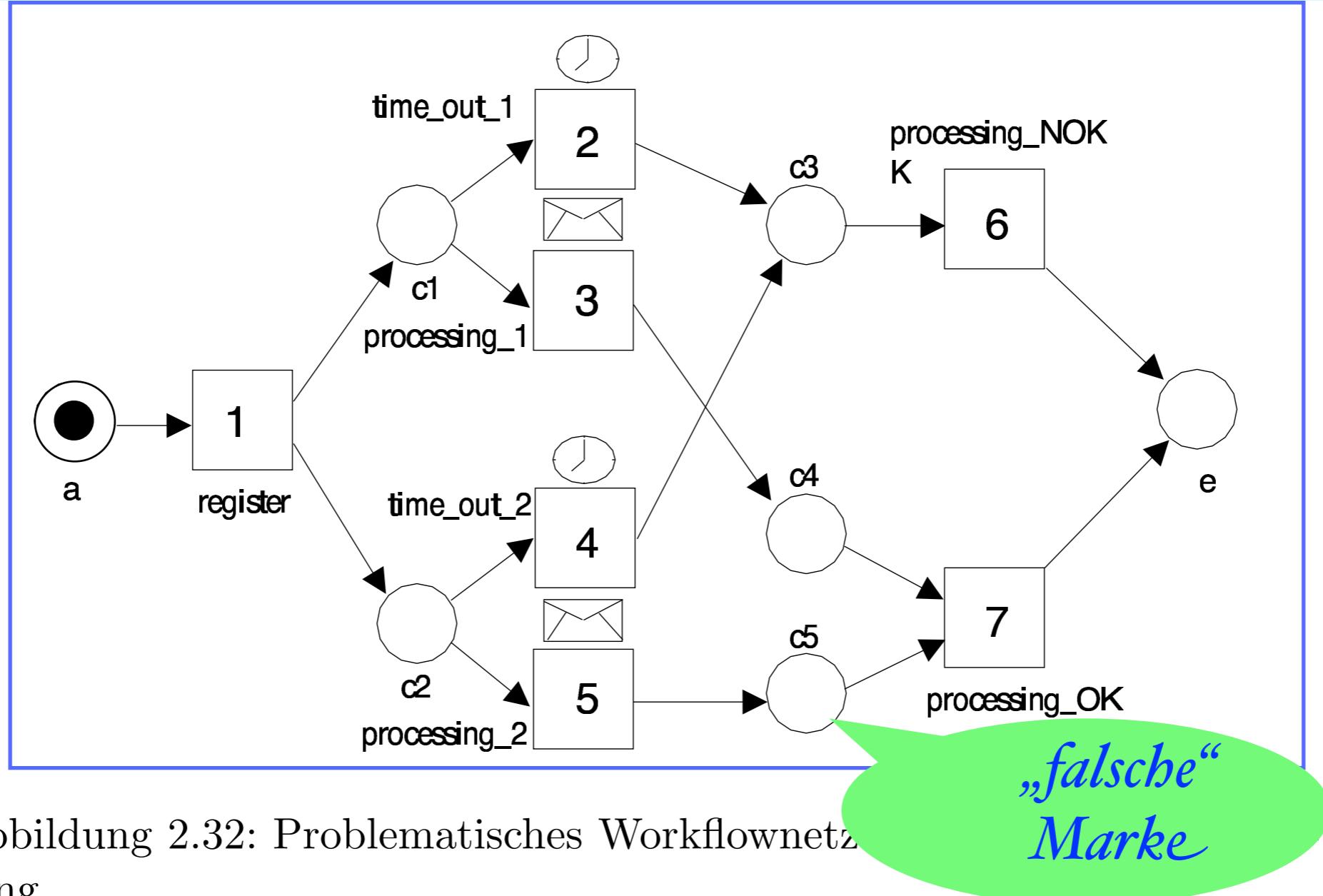


Abbildung 2.32: Problematisches Workflownetz
tung

Beispiel: Problematischer Workflow

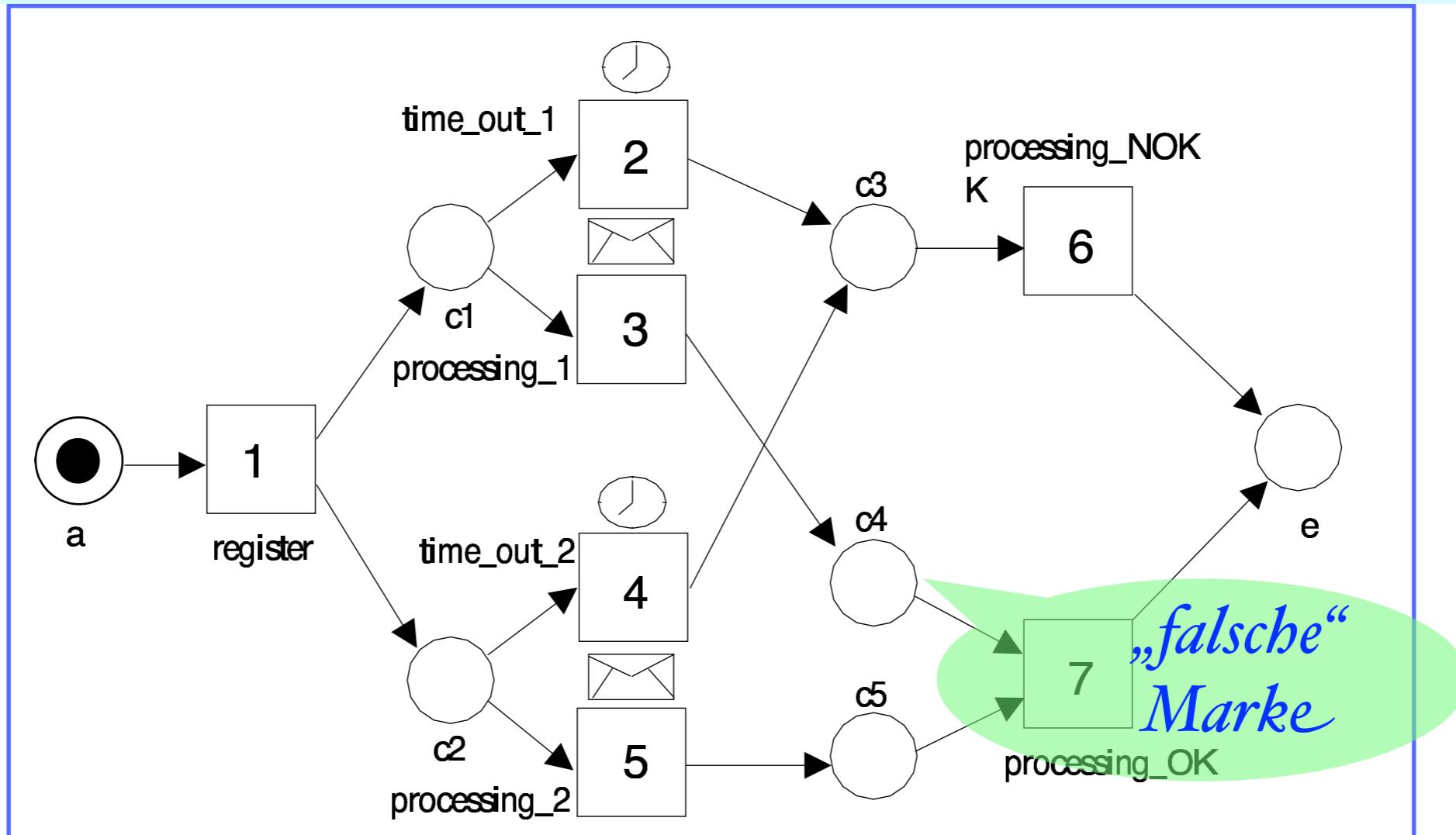


Abbildung 2.32: Problematisches Workflownetz für Beschwerdebearbeitung

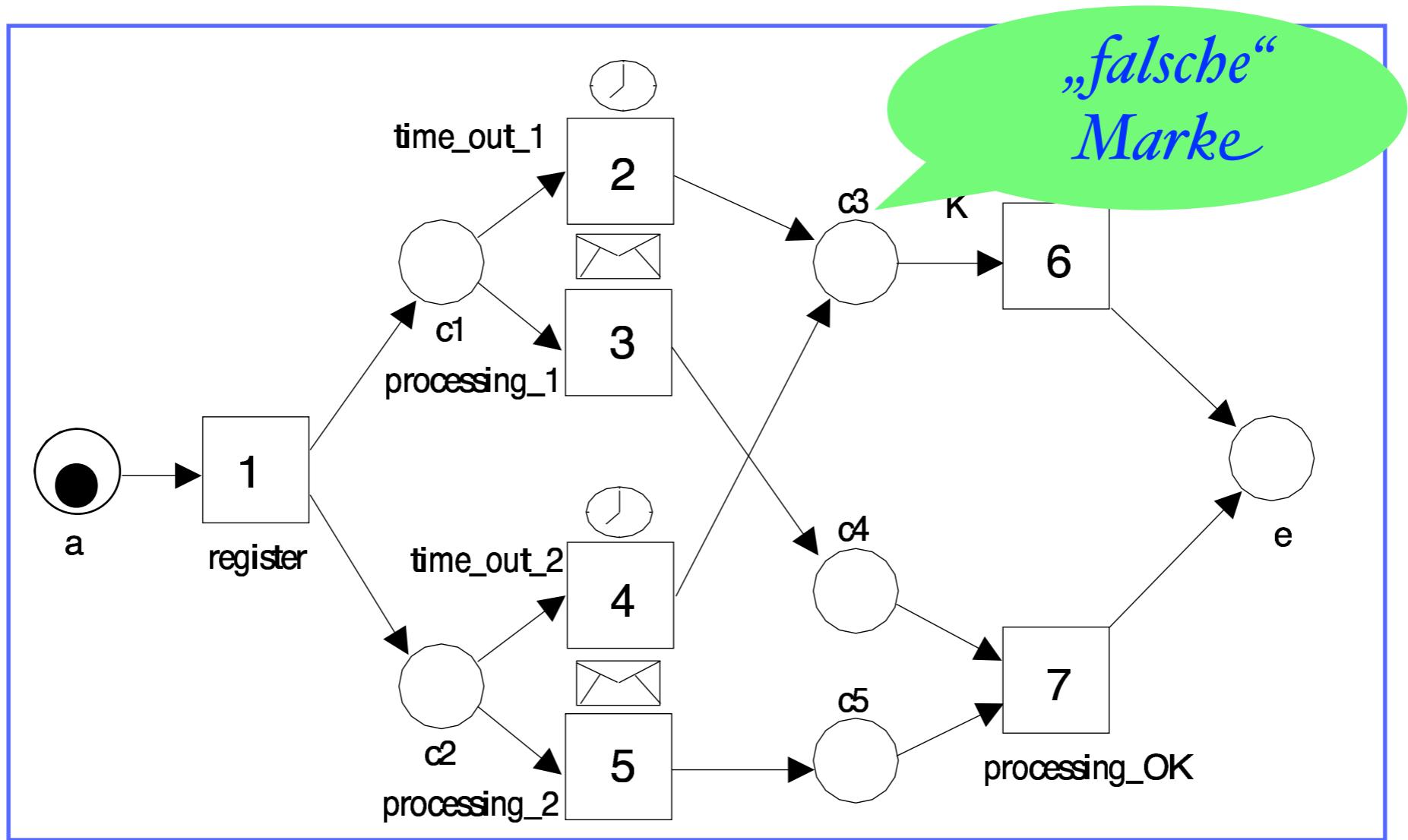


Abbildung 2.32: Problematisches Workflownetz für Beschwerdebearbeitung

Korrekte WF-Netz

Definition 2.18 Ein WF-Netz $\mathcal{N} = (P, T, F, \mathbf{m}_a)$ heißt korrekt, falls gilt :

- a) $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) \exists w \in T^* : \mathbf{m} \xrightarrow{w} \mathbf{m}_e$
- b) $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m}(e) \geq 1 \Rightarrow \mathbf{m} = \mathbf{m}_e$
- c) $\forall t \in T \exists \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m} \xrightarrow{t}$

- a) Aus jeder erreichbaren Markierung ist eine ordnungsgemäße Termination möglich.
- b) Genau eine Marke in dem Endplatz e ist die einzige Möglichkeit zu terminieren.
- c) Jede Transition kann in einer möglichen Schaltfolge schalten, denn sonst wäre sie nutzlos.

Nachweise der Korrektheit

WF-Netz \mathcal{N} korrekt,

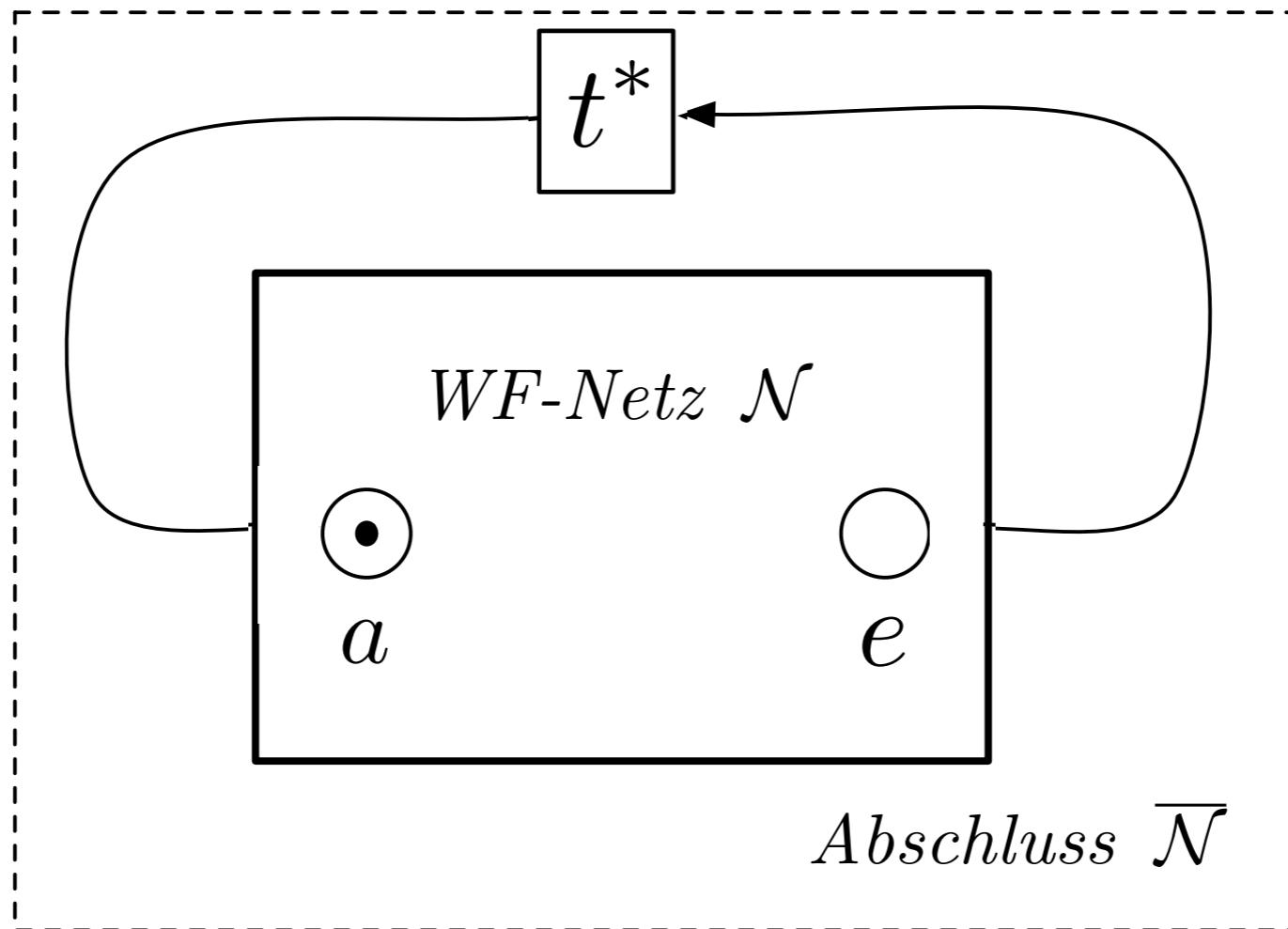


$\overline{\mathcal{N}}$ lebendig und beschränkt

*Jede Transition
behält die Eigenschaft
potenziell zu schalten.*

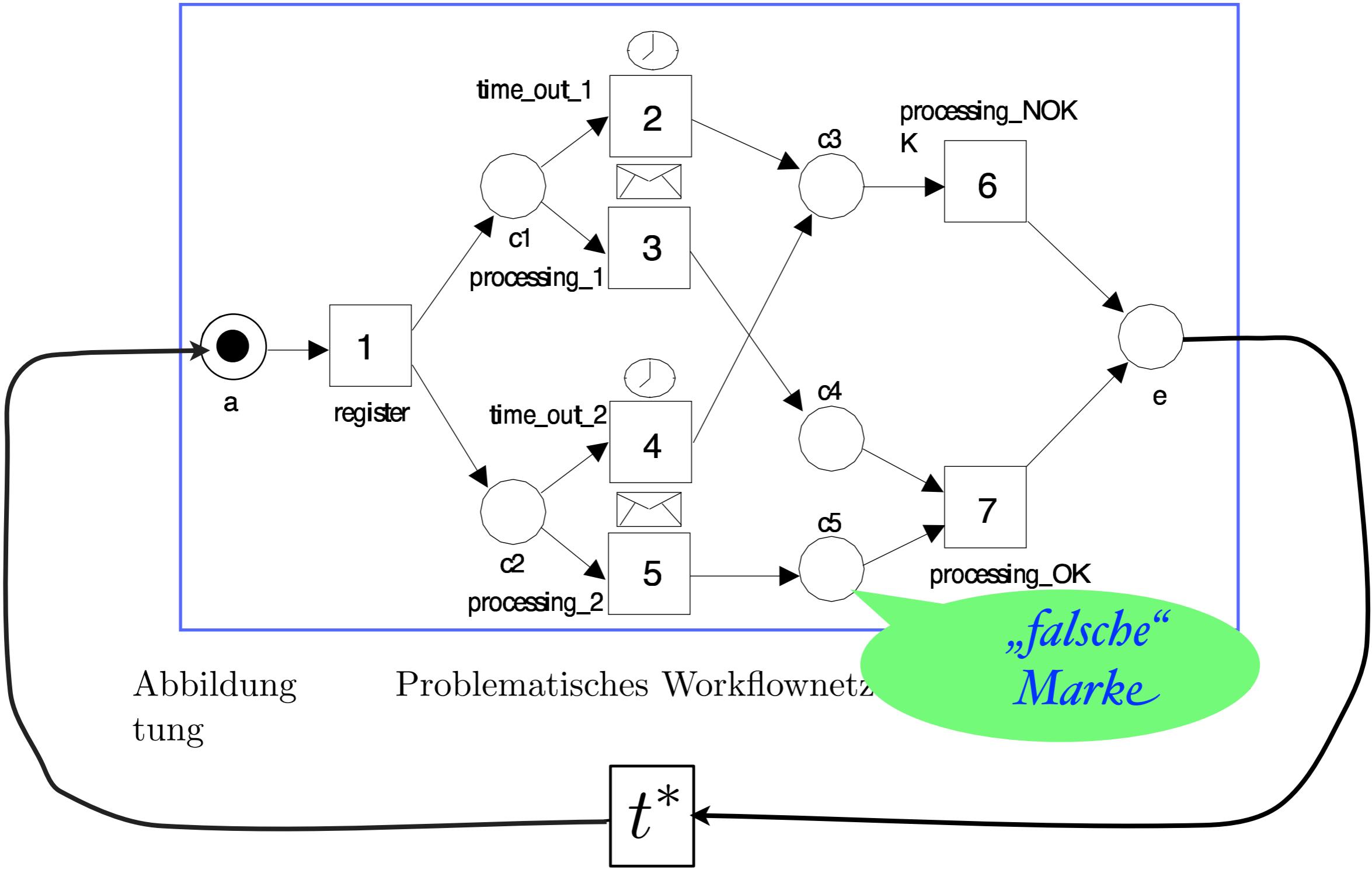
*Es gibt eine obere
Schranke für die
Markenzahl*

*Eigenschaften, für die
Algorithmen bestehen.*

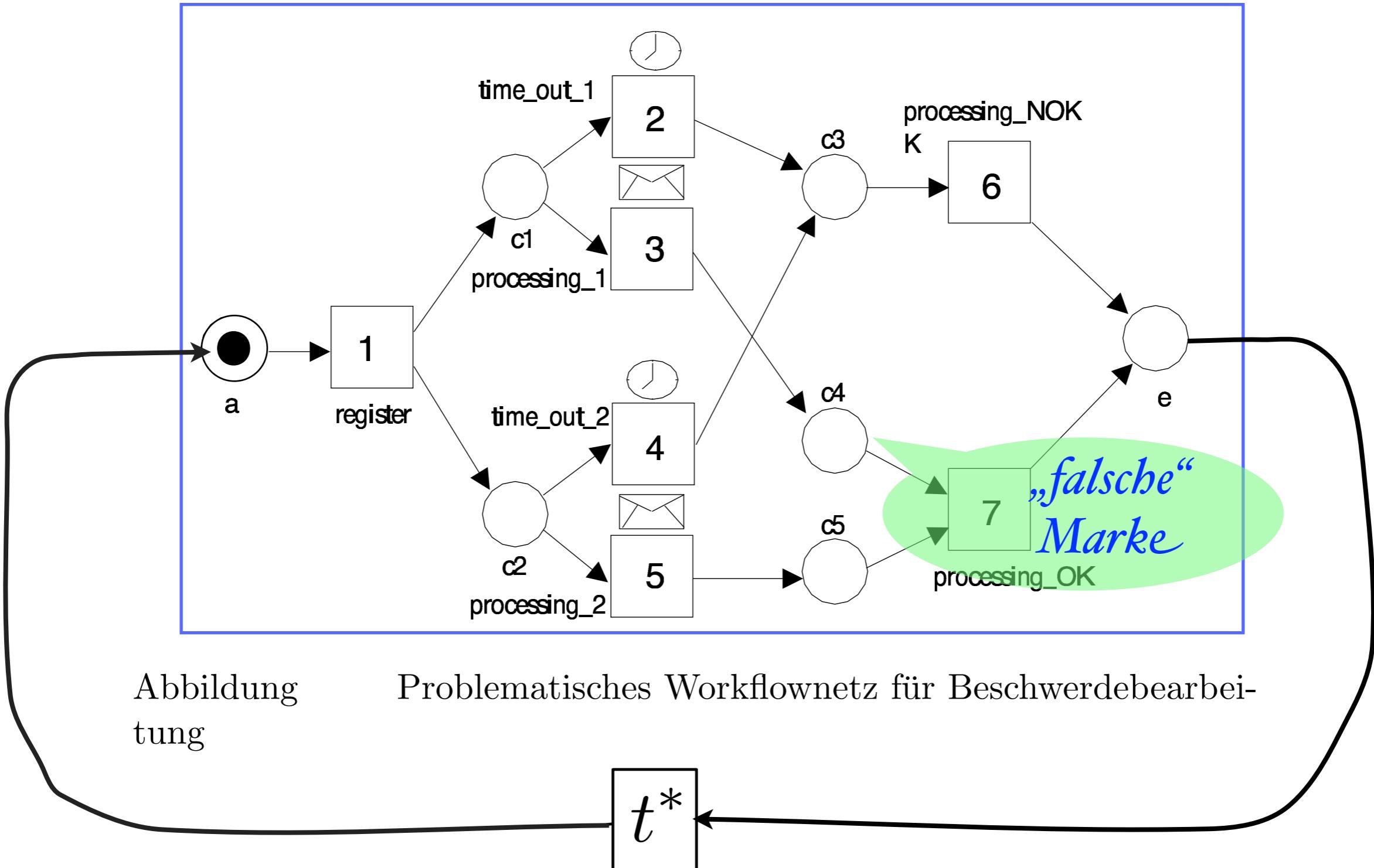


Satz Ein WF-Netz \mathcal{N} ist genau dann korrekt, wenn sein Abschluss $\overline{\mathcal{N}}$ lebendig und beschränkt ist.

Es handelt sich um die Eigenschaften „beschränkt“ und „lebendig“, die in der Tabelle 3.1 auf Seite 85 definiert werden und zu denen im selben Kapitel Entscheidungsalgorithmen entwickelt



?



?

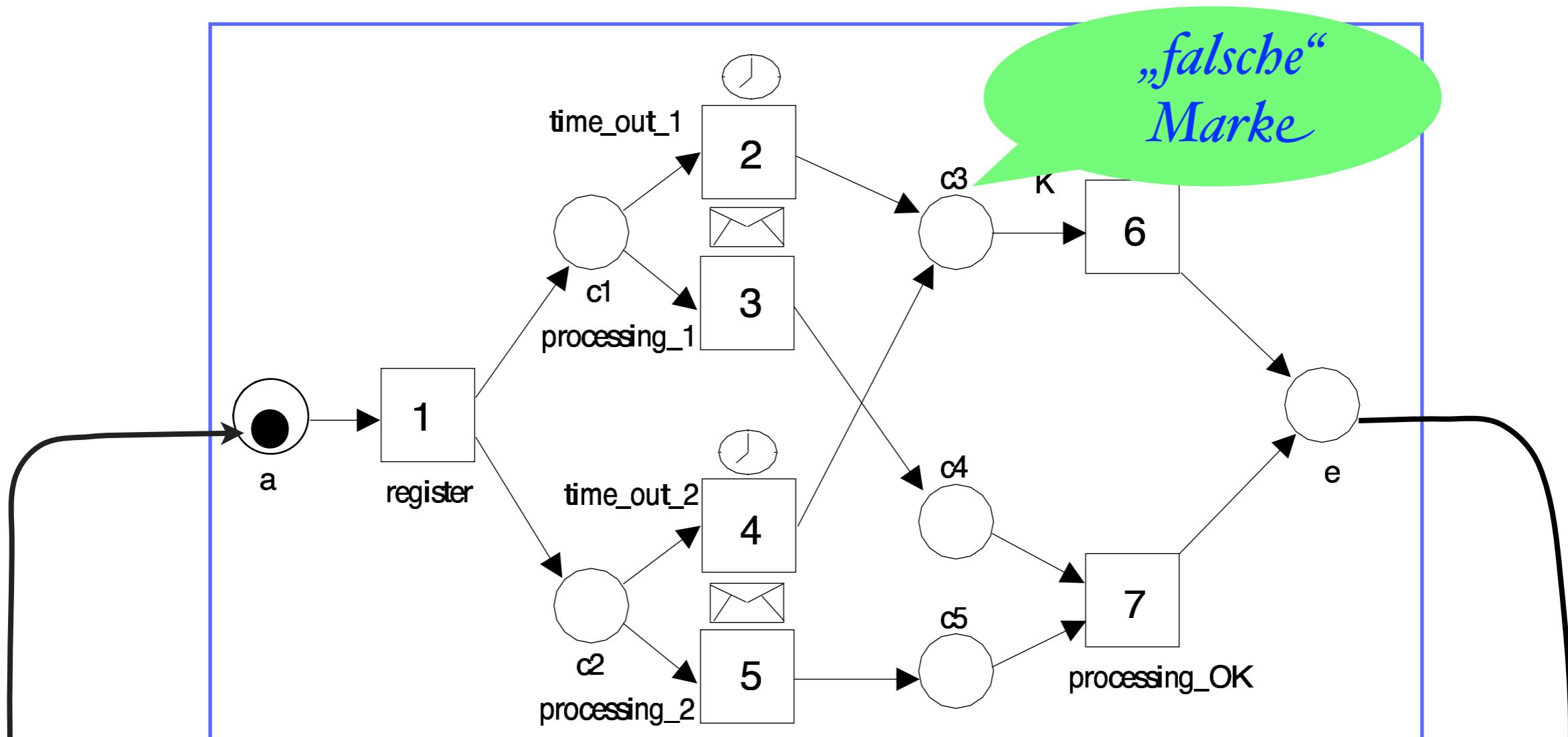


Abbildung
tung

Problematisches Workflownetz für Beschwerdebearbei-

t^*

?

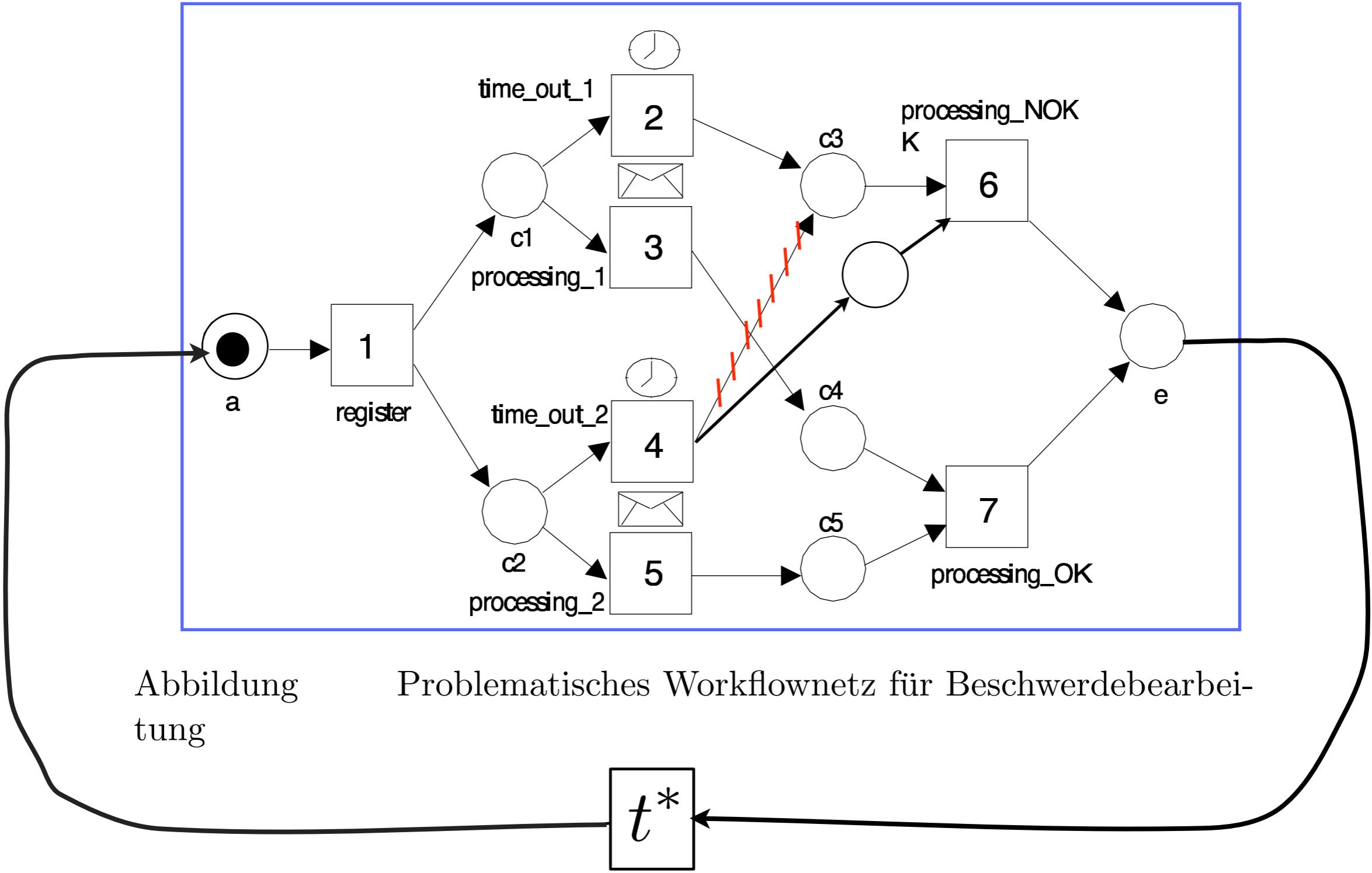


Abbildung
tung

Problematisches Workflownetz für Beschwerdebearbei-

t^*

?

FGI 2

Daniel Moldt

Bankiersproblem

Das Bankiers-Problem

• Bankiers-Problem

1 000 000 €

*Betriebsmittel-
Vergabe-Problem*
*resource-
allocation-problem*



Ein **Bankier** besitzt ein **Kapital g.**

Seine **n Kunden** erhalten wechselnden Kredit. Jeder Kunde muss seinen maximalen Kreditwunsch von vorneherein bekanntgeben und wird nur als Kunde akzeptiert, wenn dieser das Kapital nicht übersteigt.

Das Bankiers-Problem

Ein **Bankier** besitzt ein **Kapital** g .

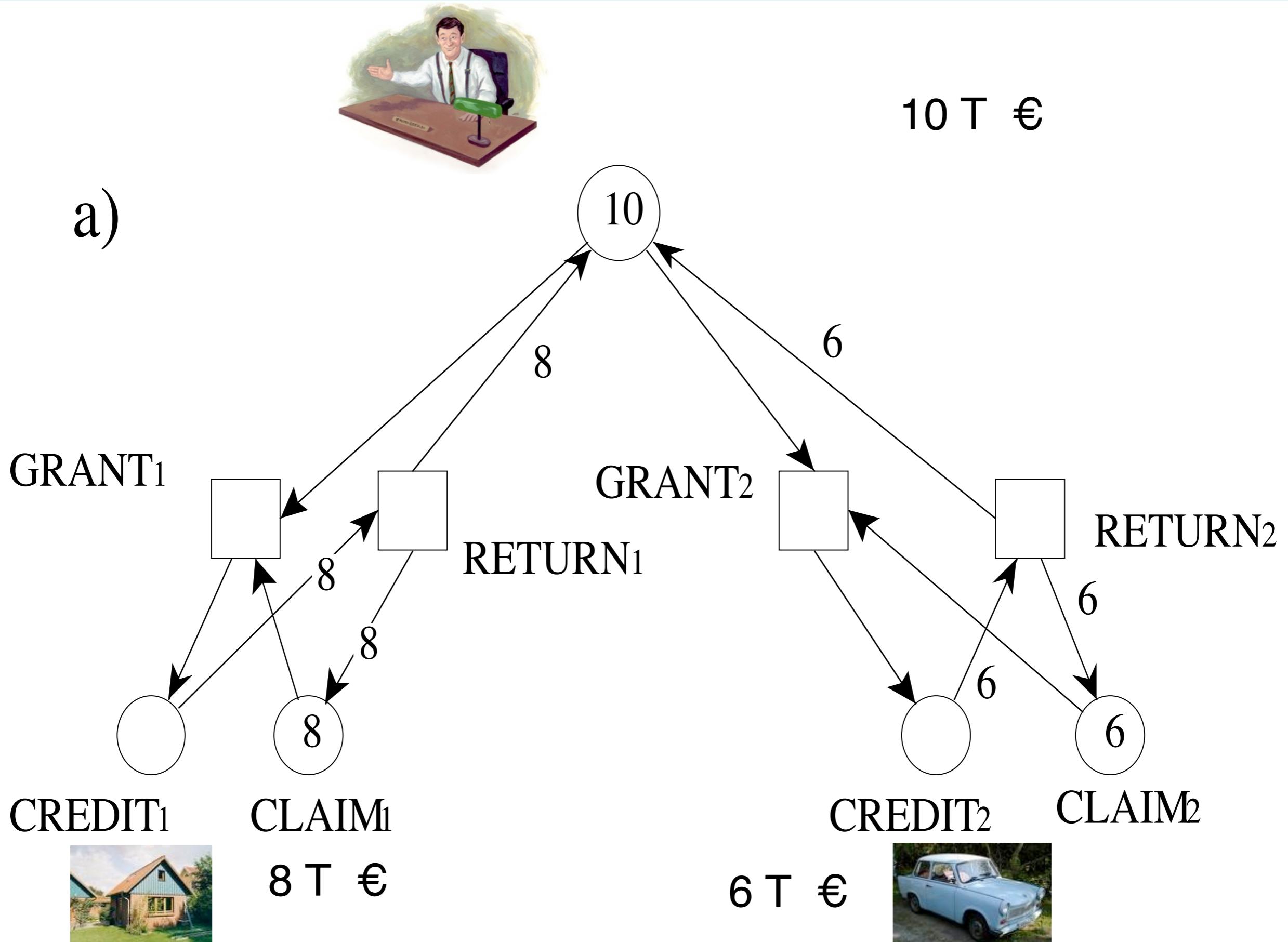
Seine **n Kunden** erhalten wechselnden Kredit. Jeder Kunde muss seinen maximalen Kreditwunsch von vorneherein bekanntgeben und wird nur als Kunde akzeptiert, wenn dieser das Kapital nicht übersteigt.

Kredite werden nicht entzogen. Dafür muss aber jeder Kunde versprechen, den Maximalkredit auf einmal nach endlicher Zeit zurückzuzahlen.

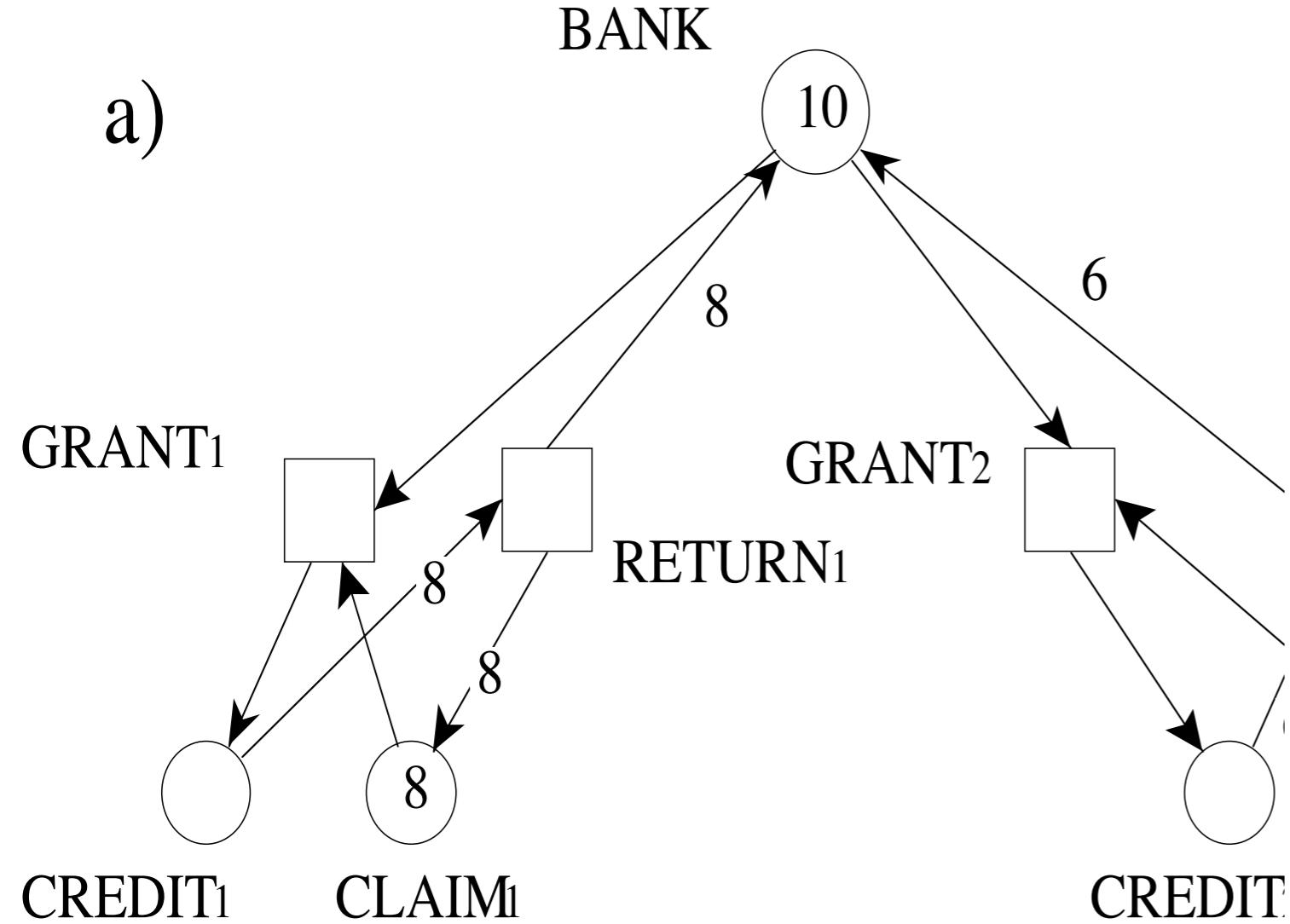
Der Bankier verspricht, jede Bitte um Kredit in endlicher Zeit zu erfüllen.

Für den Bankier besteht natürlich das Problem der **Verklemmung**: es kann sein, dass mehrere Kunden noch nicht ihren Maximalkredit erhalten haben, das Restkapital des Bankiers aber zu klein ist, mindestens einen Kunden total zu befriedigen, um dann nach einer Frist wieder neues Kapital zu haben.

Bankiersproblem als Petrinetz



a)



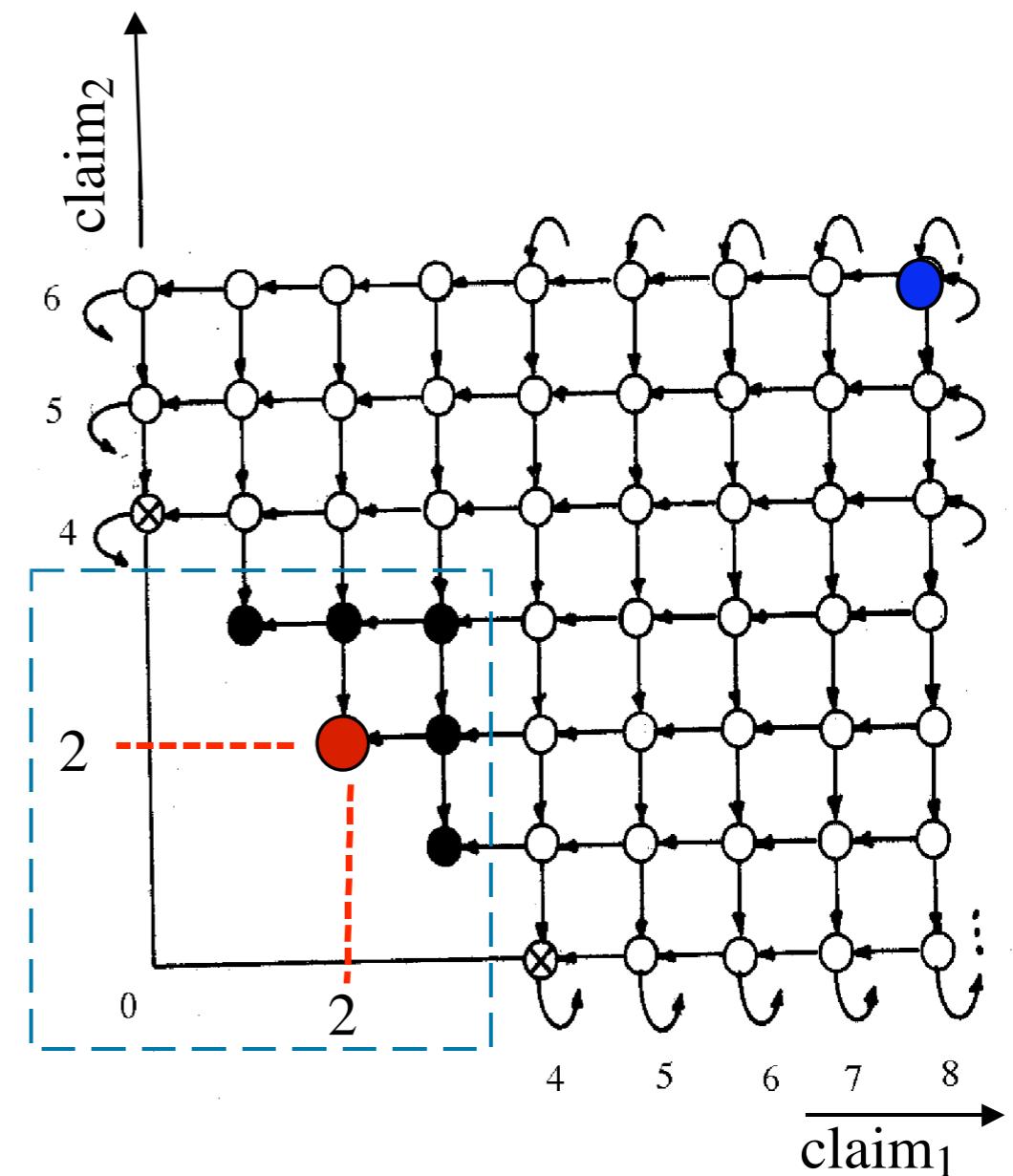
$$5 - 3 = 2$$

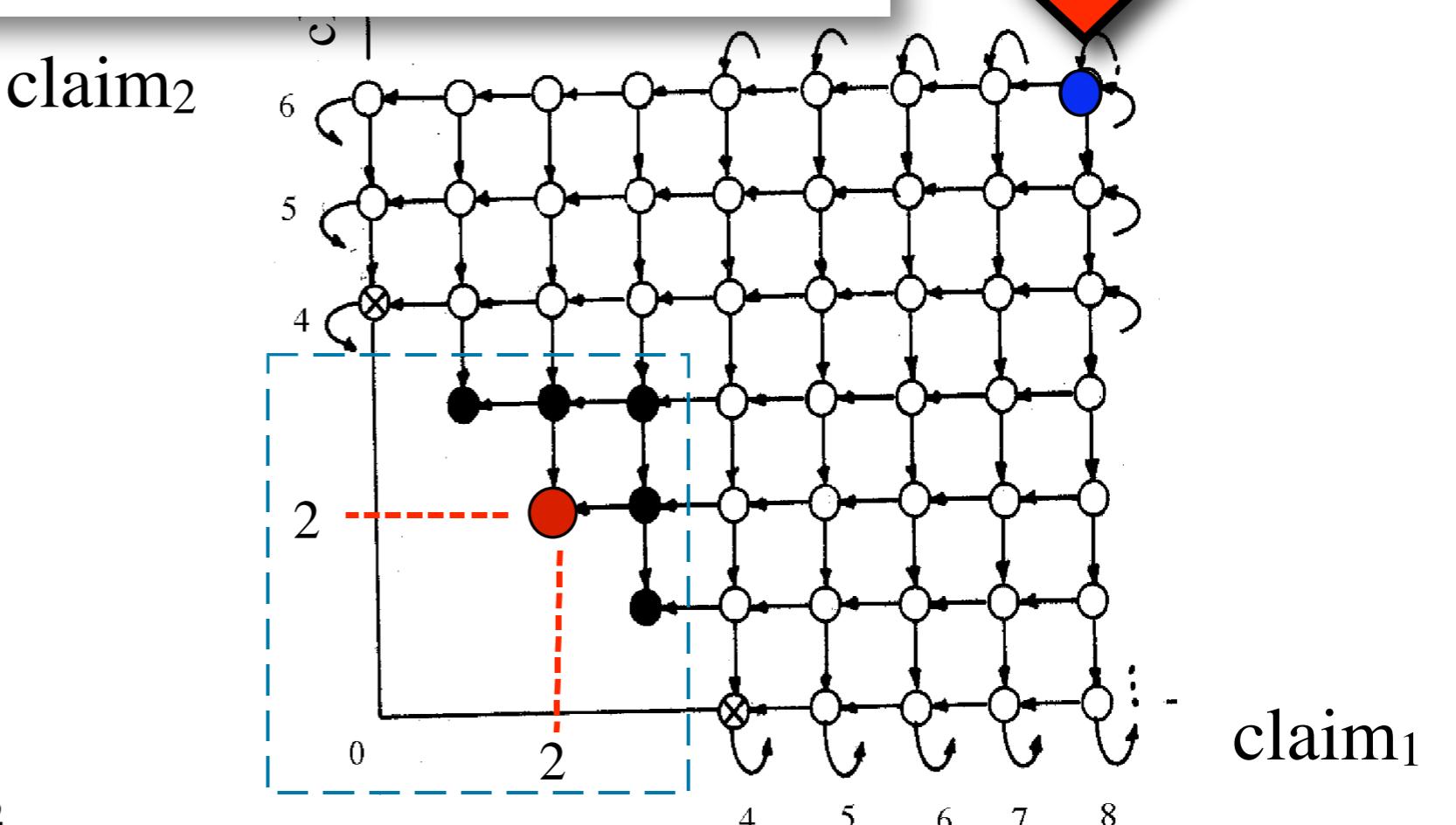
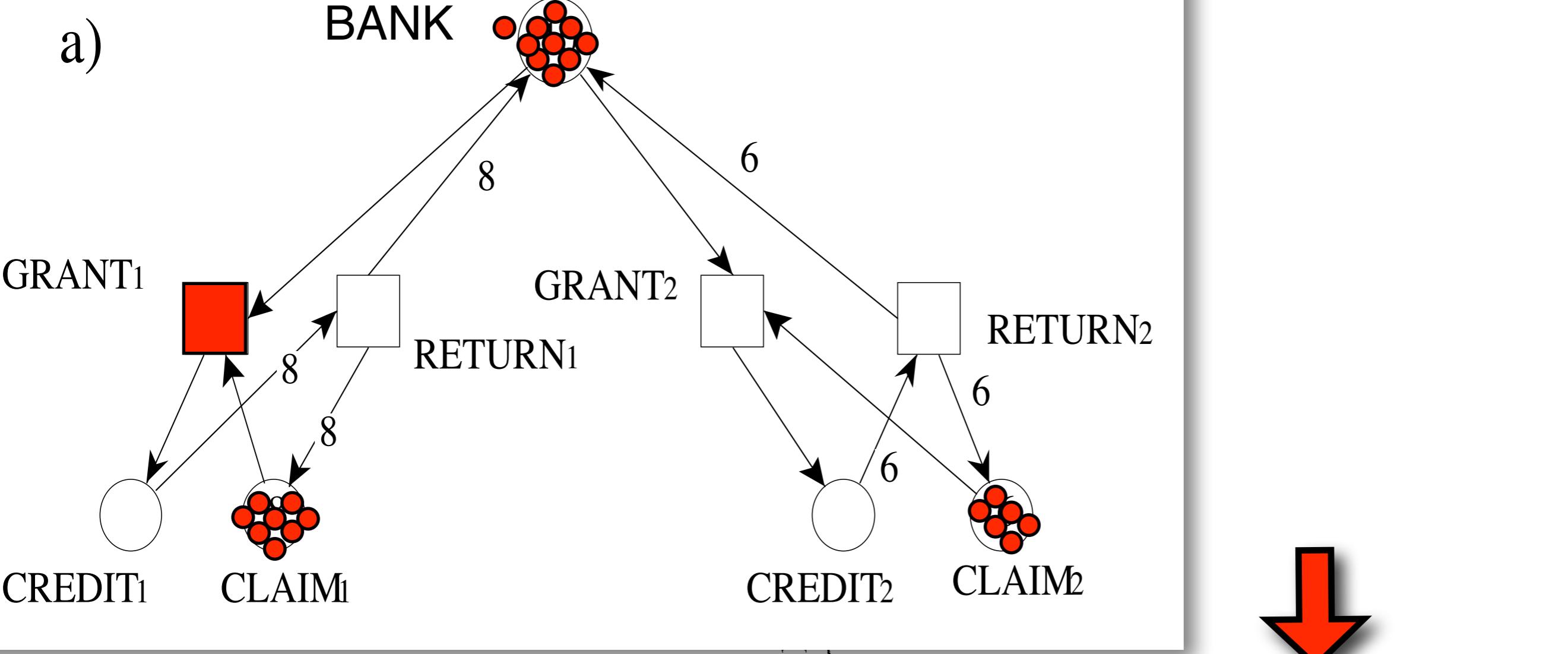
$$\forall m \in R(\mathcal{N})$$

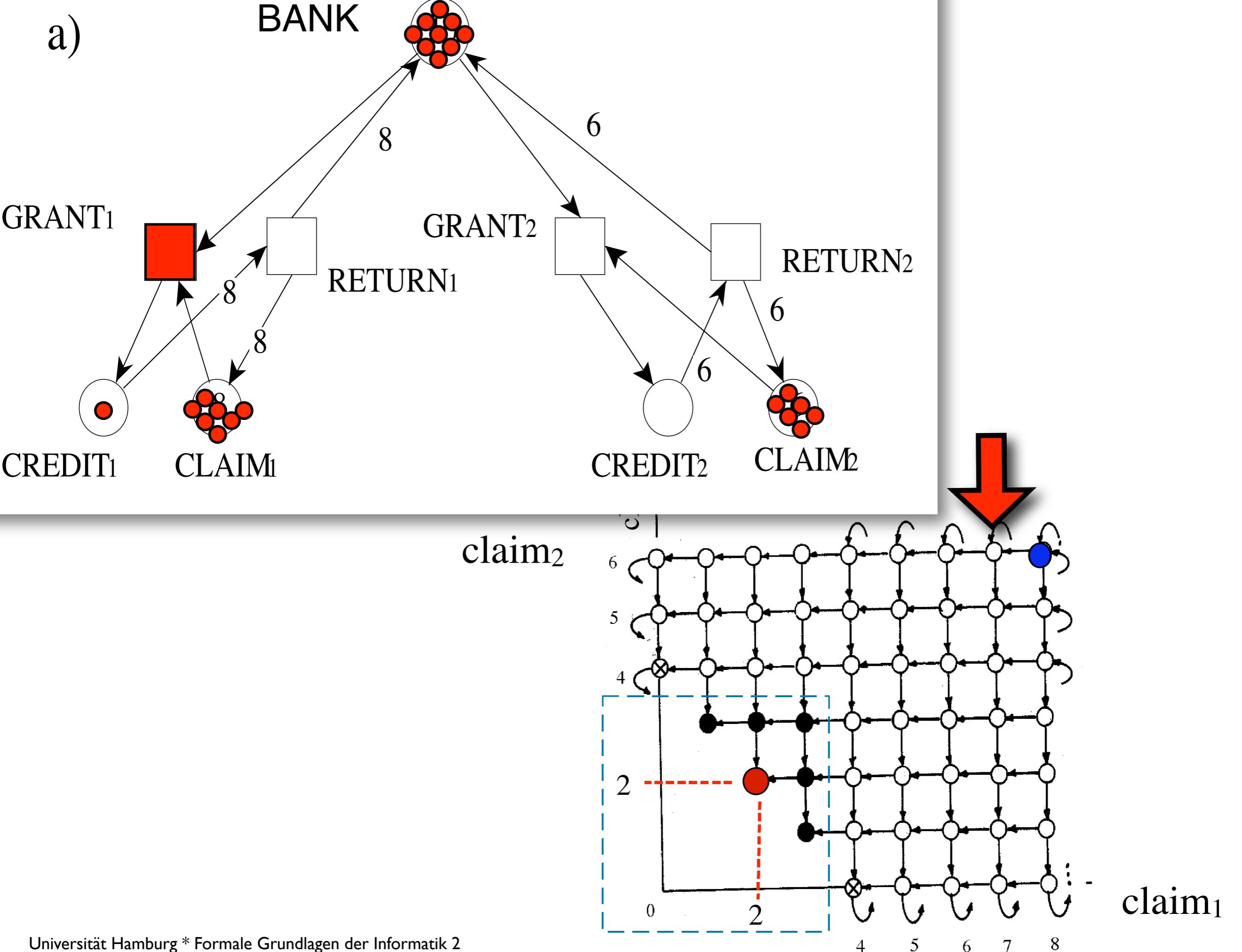
$$m(CREDIT_1) + m(CLAIM_1) = 8$$

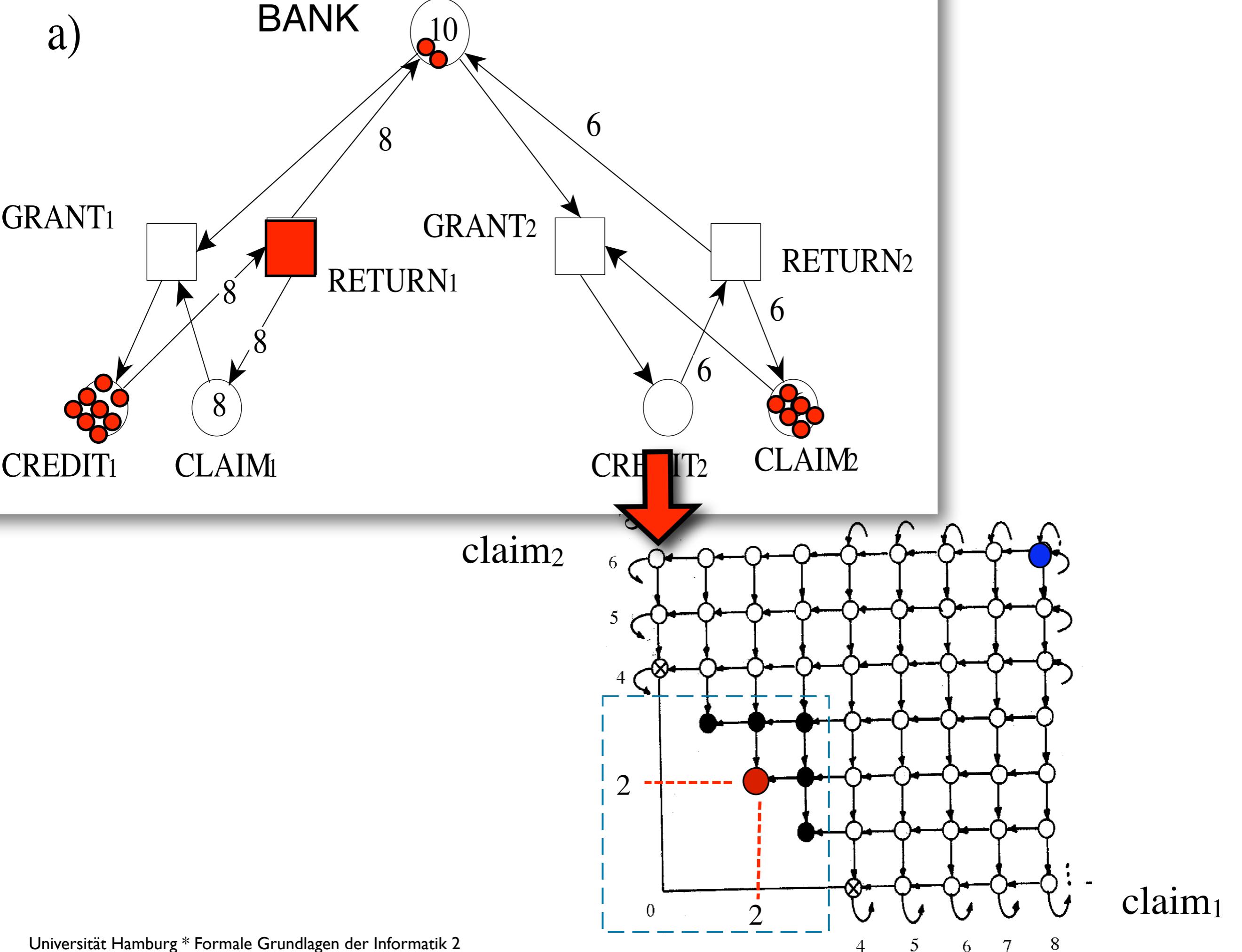
$$m(CREDIT_2) + m(CLAIM_2) = 6$$

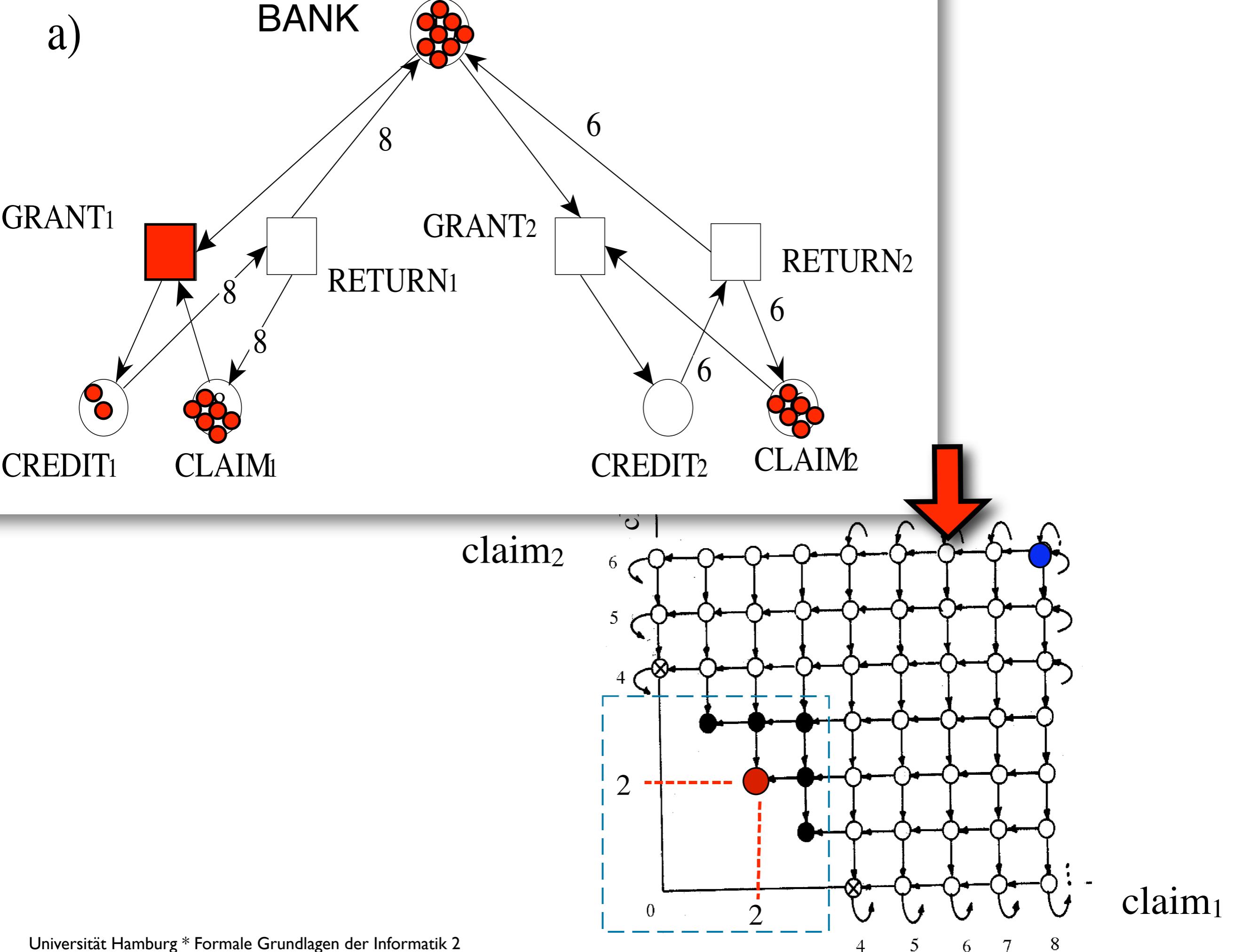
$$m(BANK) + m(CREDIT_1) + m(CREDIT_2) = 10$$

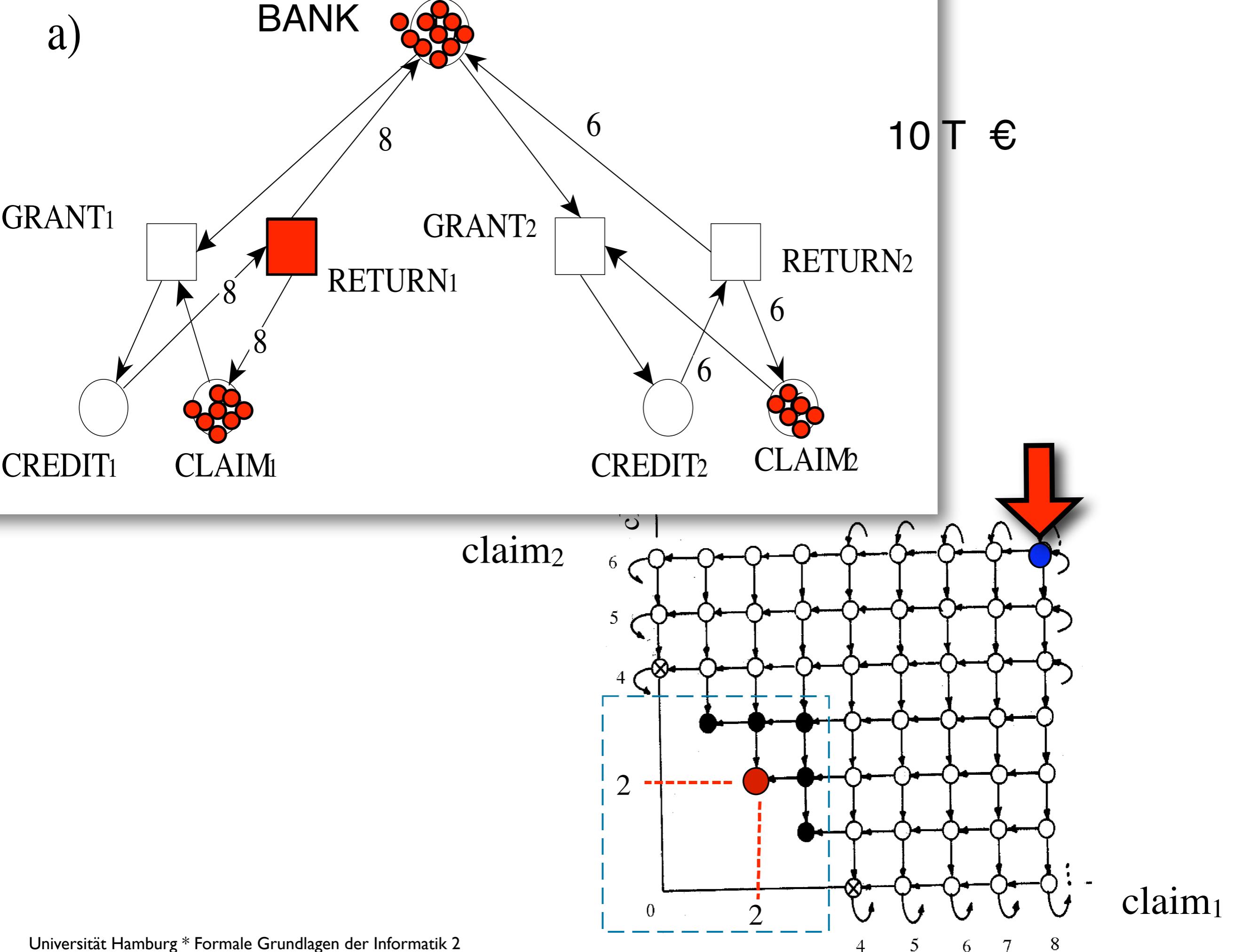


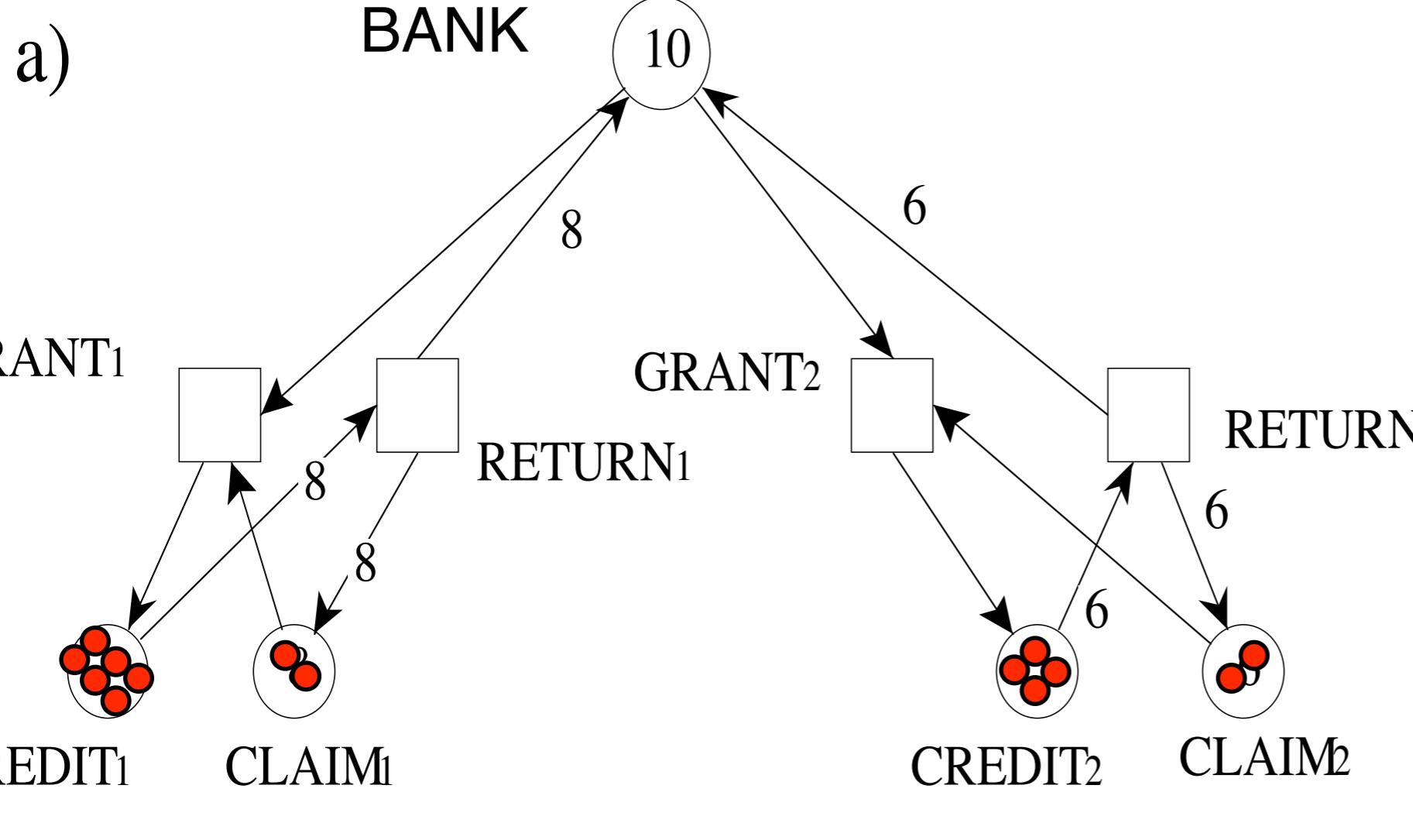






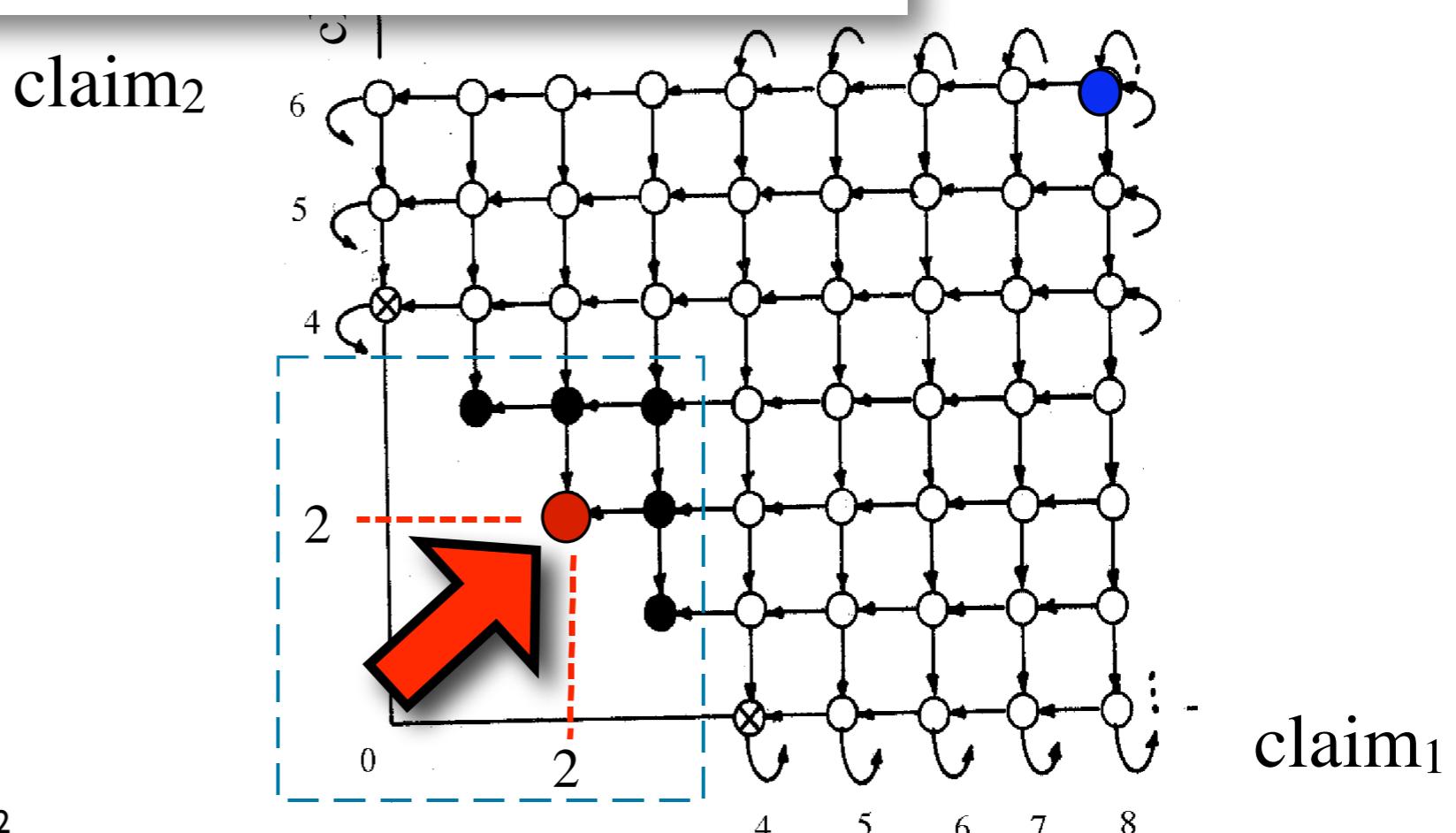


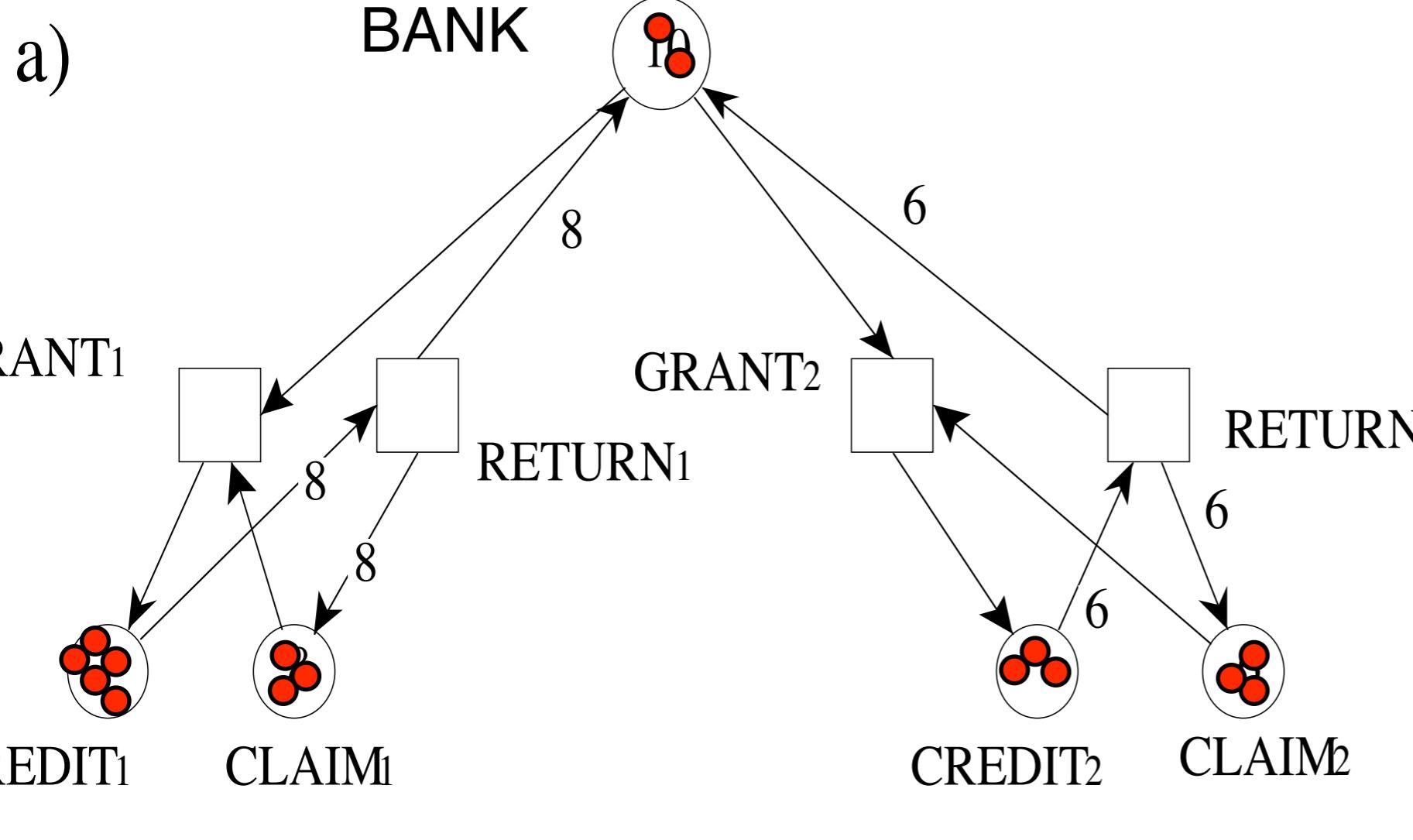




Verklemmung

deadlock

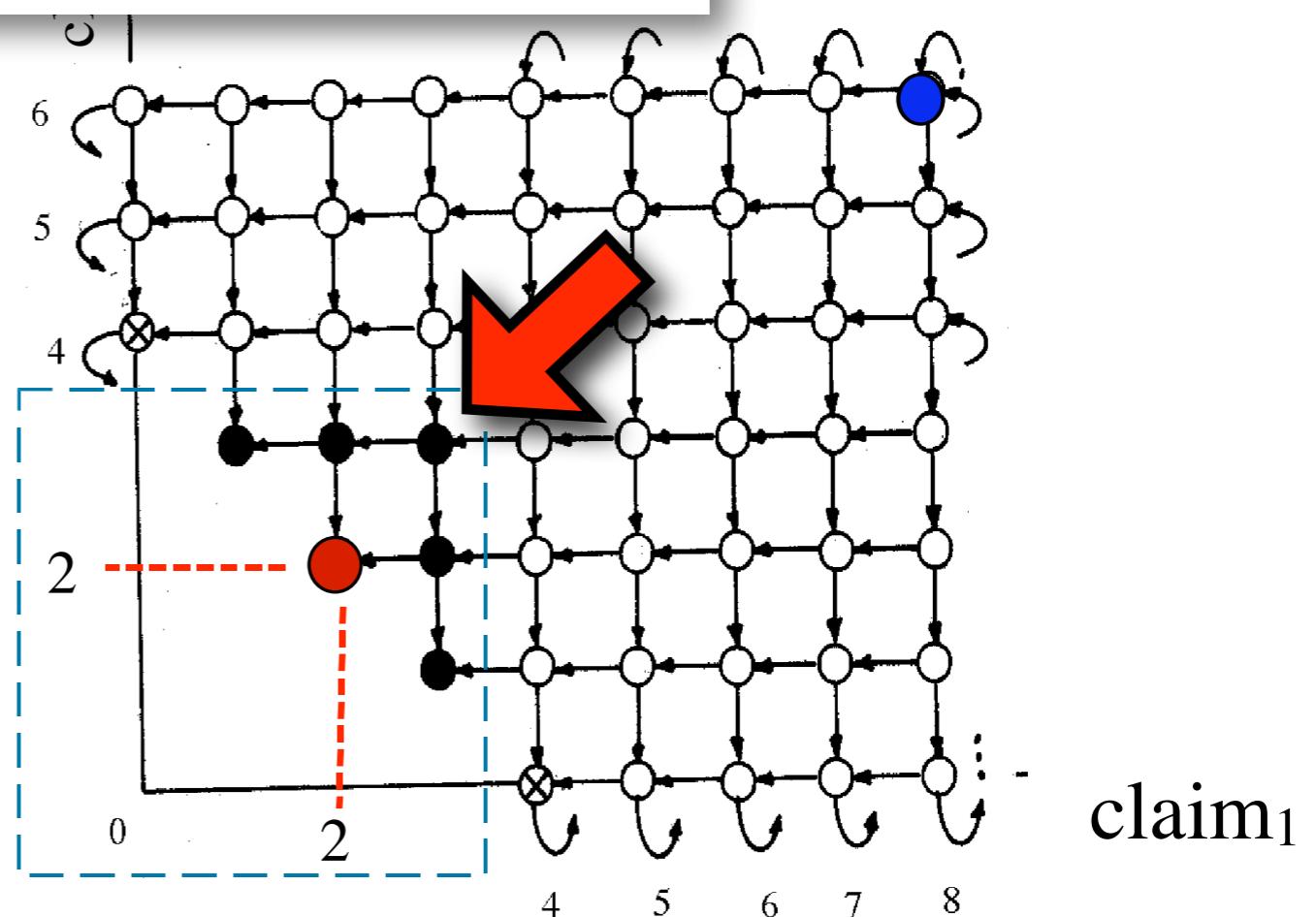




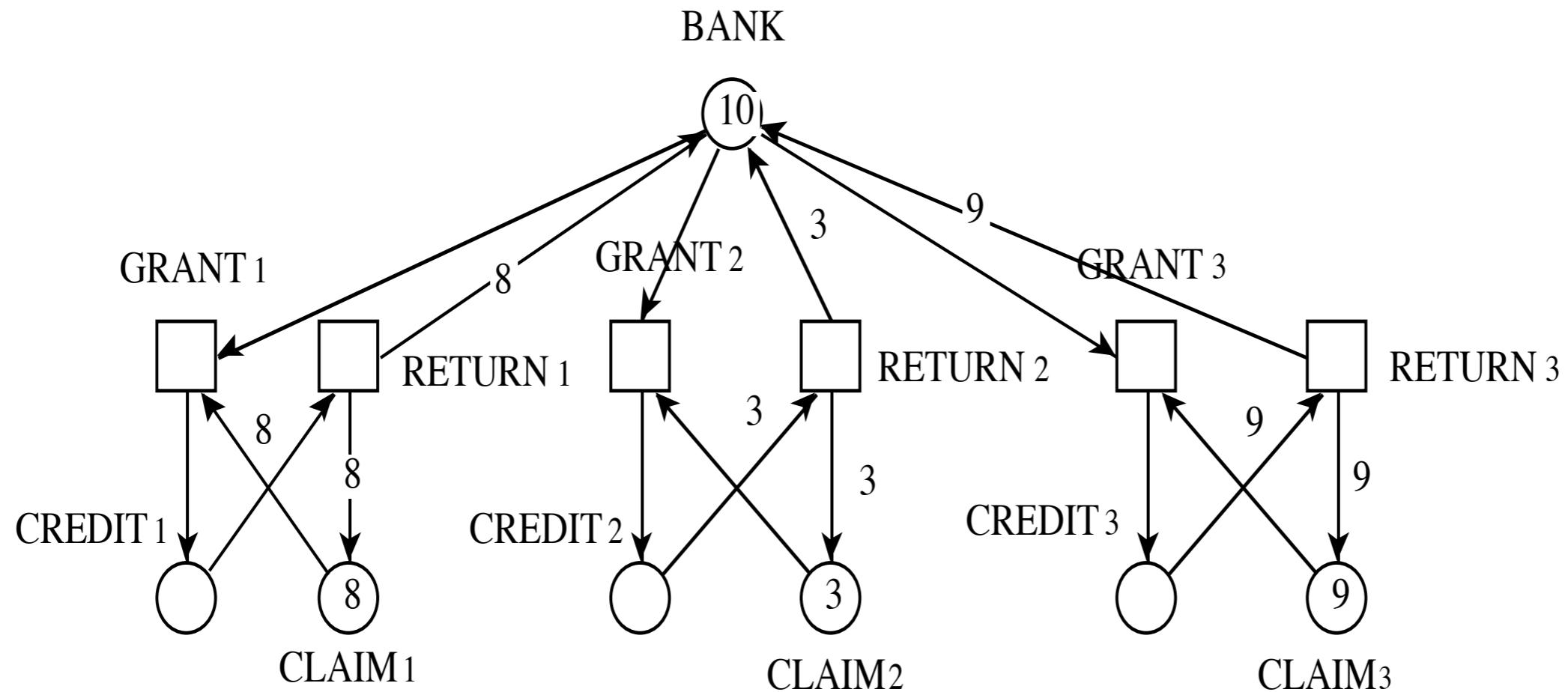
*unsicher
unsafe*

*bankers like resource
allocation
problems/algorithms*

claim₂

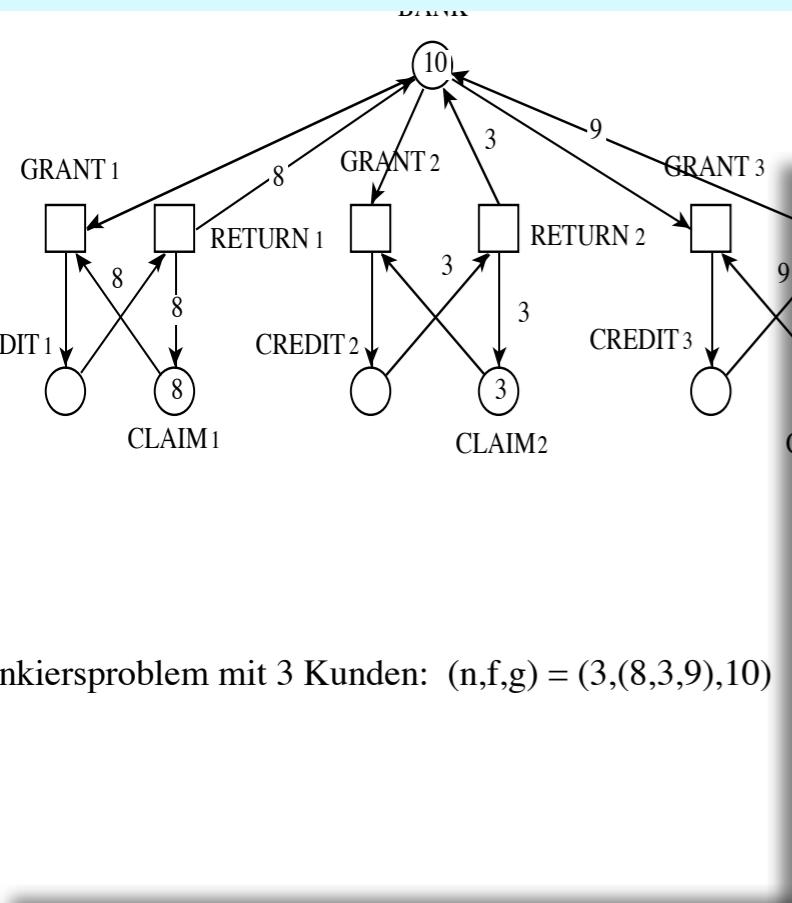


3 Kreditnehmer



Bankiersproblem mit 3 Kunden: $(n,f,g) = (3,(8,3,9),10)$

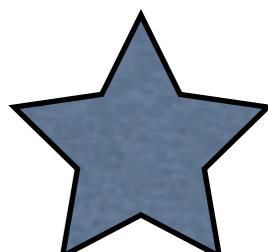
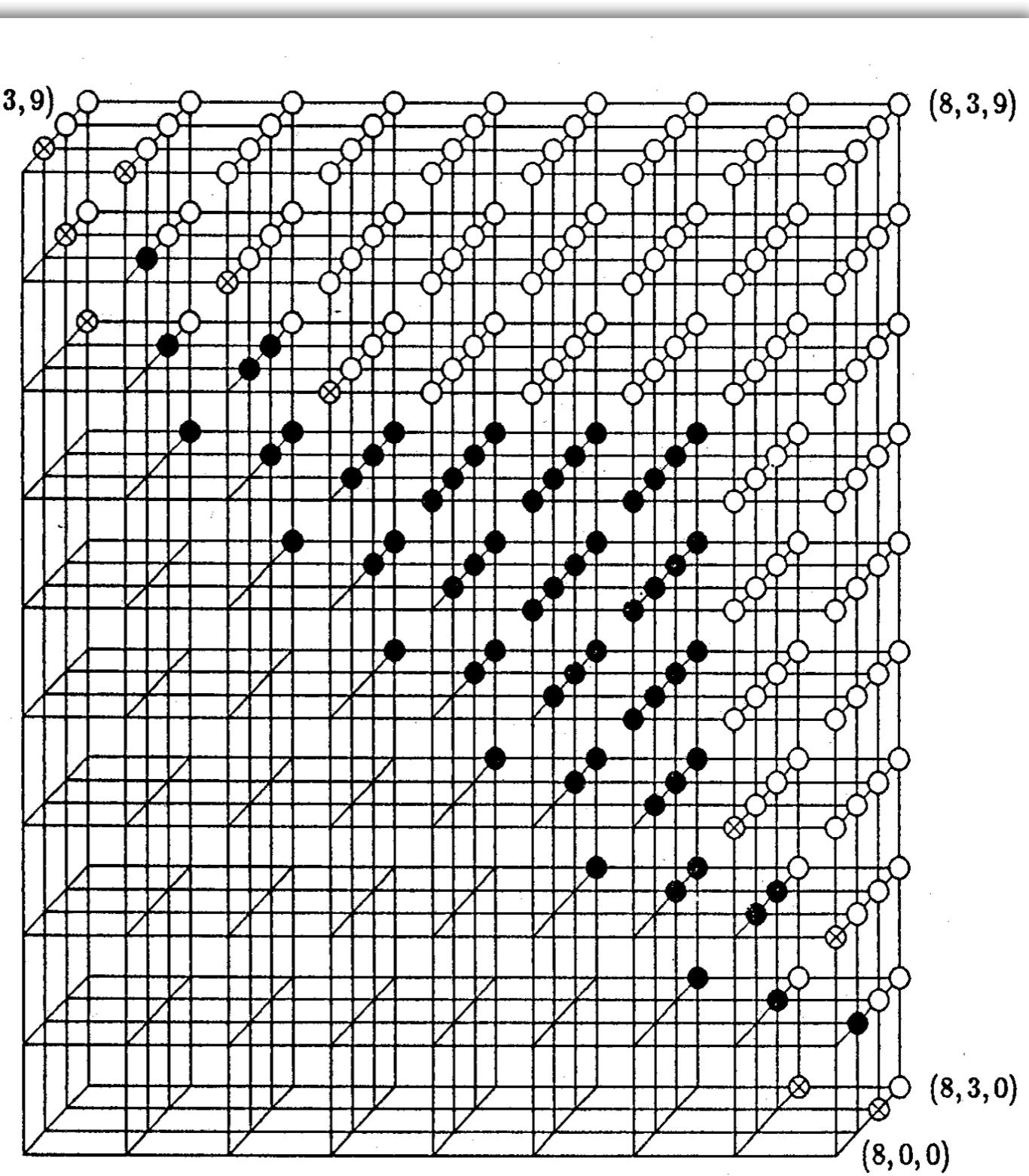
Erreichbarkeitsgraph



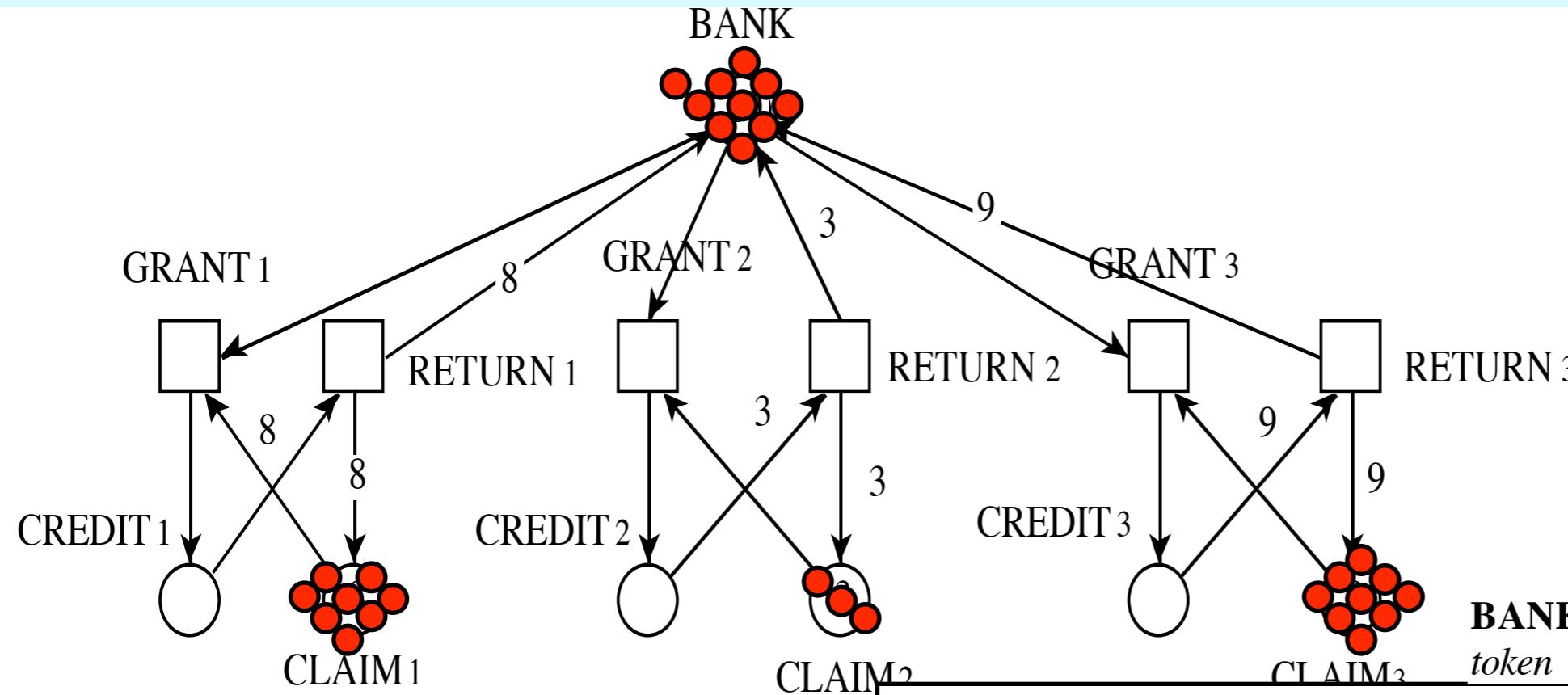
Kalkulatorsproblem mit 3 Kunden: $(n, f, g) = (3, (8, 3, 9), 10)$

195 erreichbare Markierungen
137 sichere Markierungen
58 unsichere Markierungen

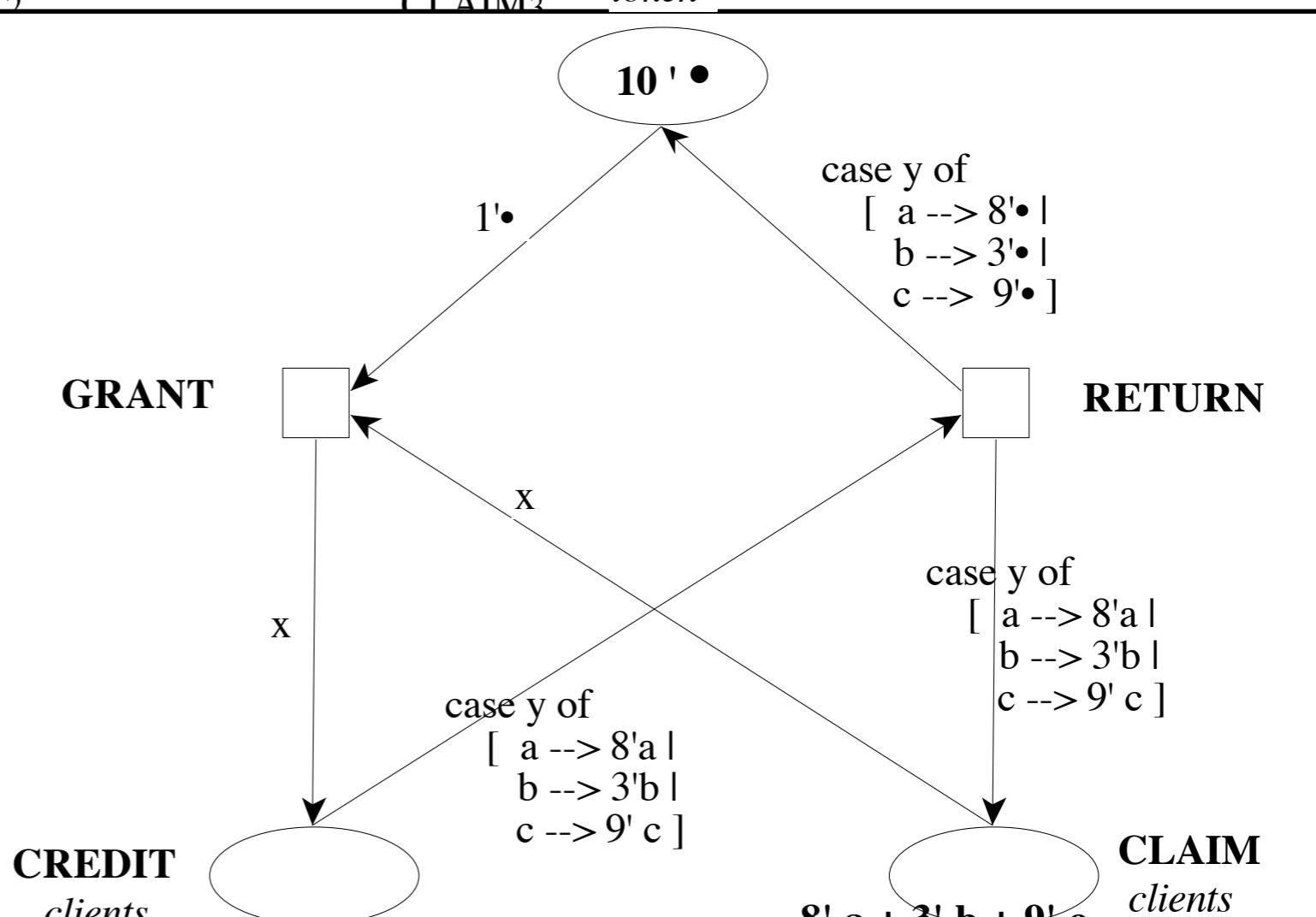
**10 minimale
 sichere Markierungen**



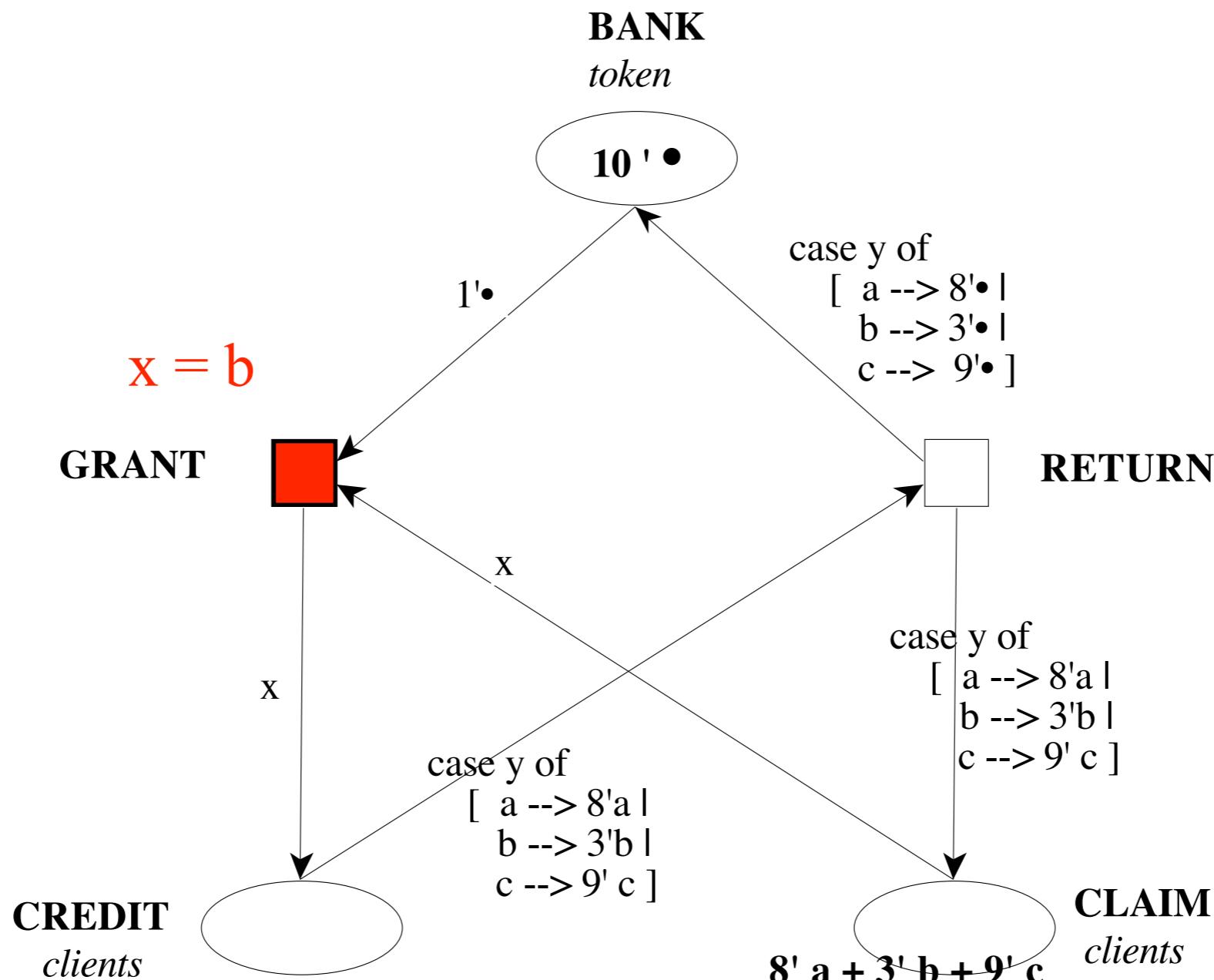
Bankiersproblem als Gefärbtes Netz



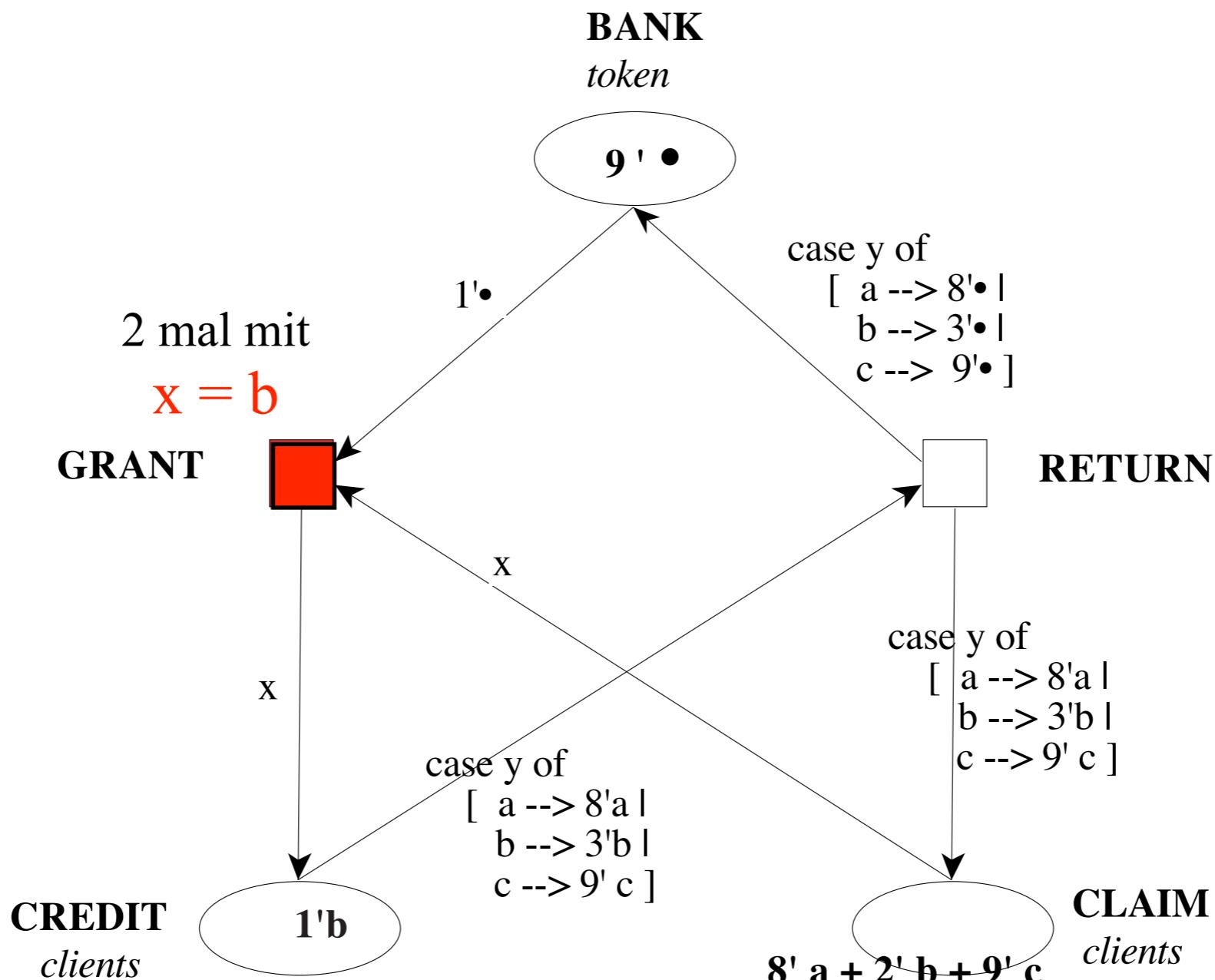
*Faltung
des
P/T-Netzes
zu einem
gefärbten
Netz*



Bankiersproblem als Gefärbtes Netz



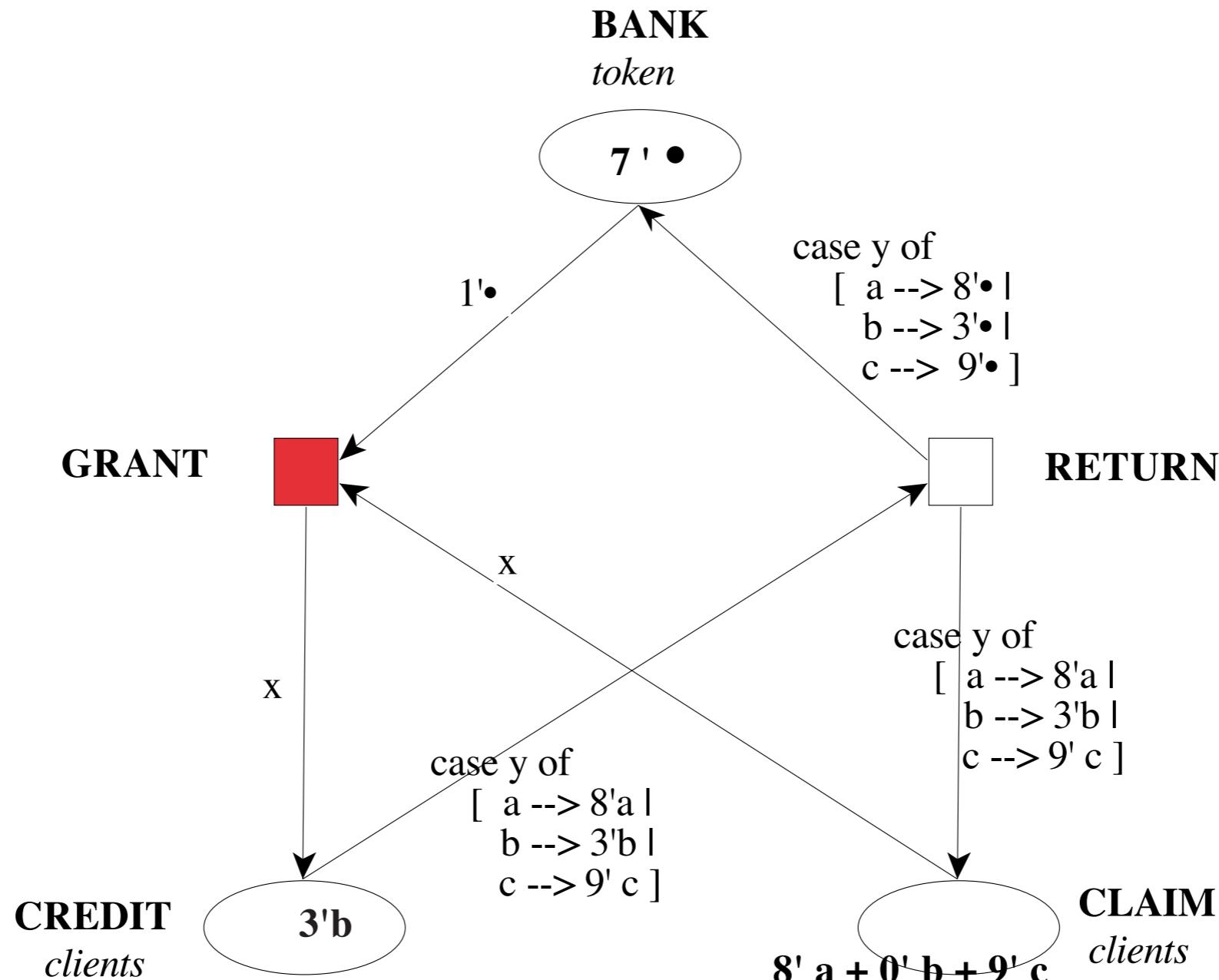
Bankiersproblem als Gefärbtes Netz



Farben: $token = \{\bullet\}$, $clients = \{a,b,c\}$

Variablen : x, y mit $dom(x) = dom(y) = clients$

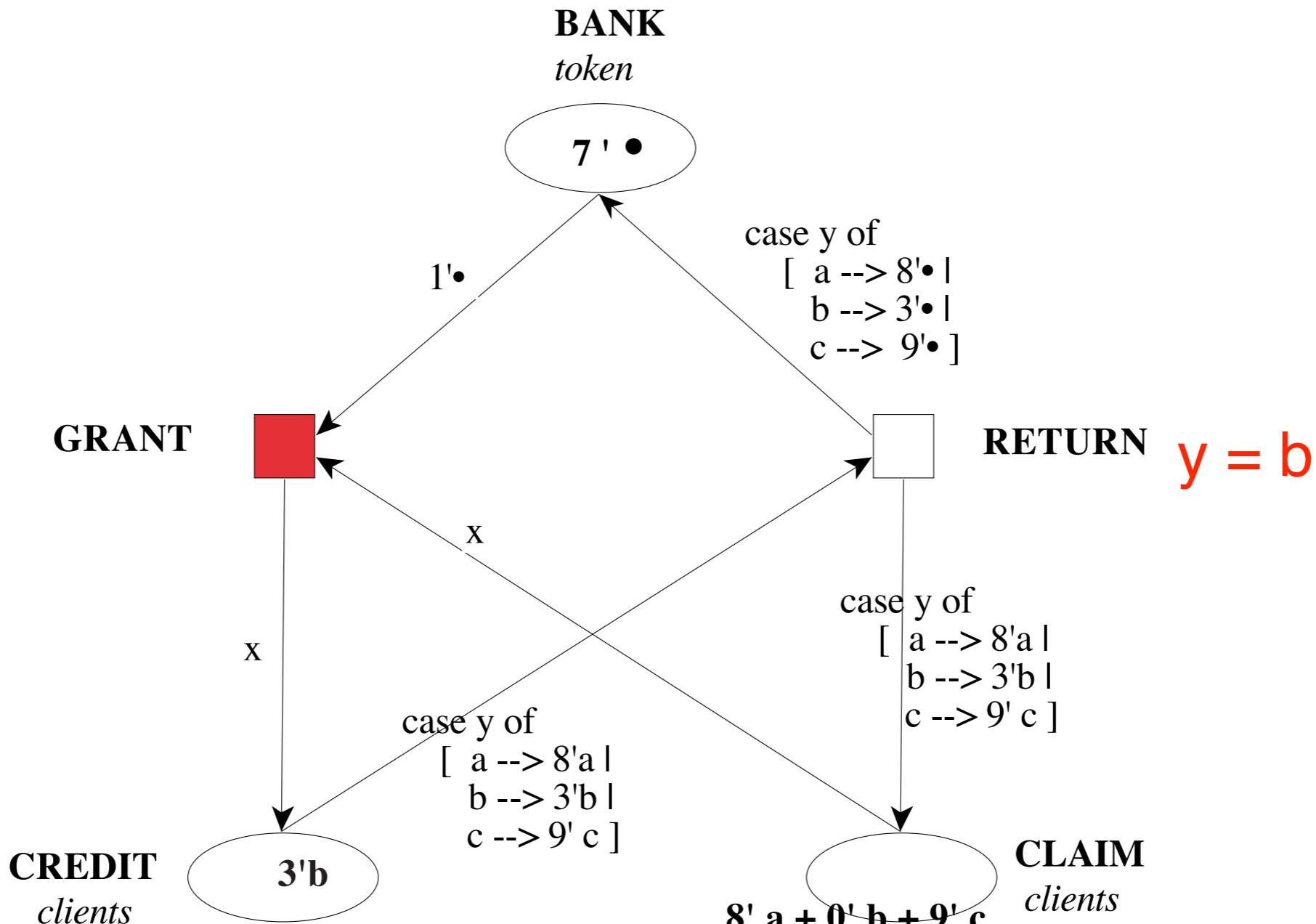
Bankiersproblem als Gefärbtes Netz



Farben: $token = \{\bullet\}$, $clients = \{a,b,c\}$

Variablen : x, y mit $dom(x) = dom(y) = clients$

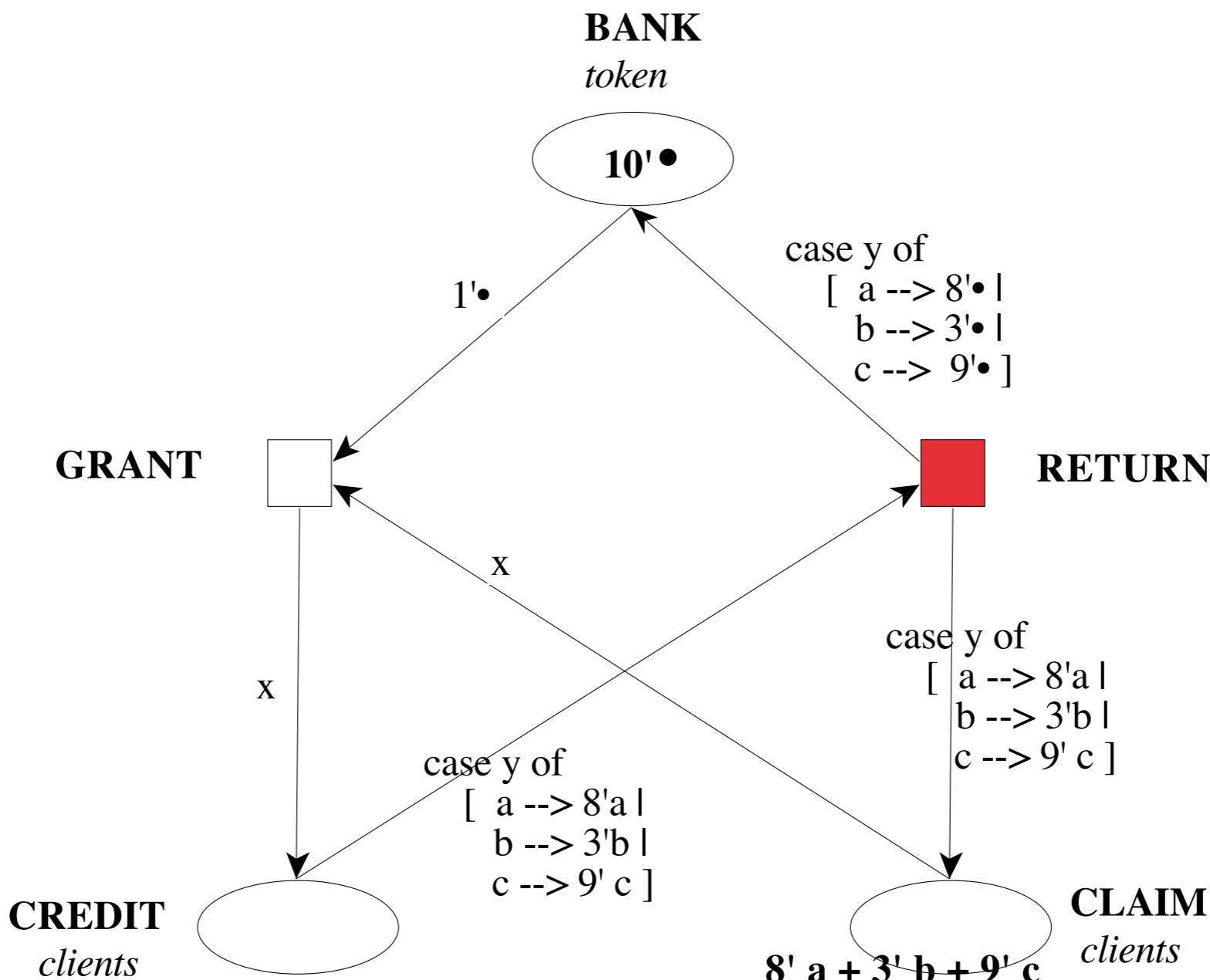
Bankiersproblem als Gefärbtes Netz



Farben: $token = \{\bullet\}$, $clients = \{a,b,c\}$

Variablen : x, y mit $\text{dom}(x) = \text{dom}(y) = clients$

Bankiersproblem als Gefärbtes Netz



Farben: $token = \{\bullet\}$, $clients = \{a,b,c\}$

Variablen : x, y mit $dom(x) = dom(y) = clients$

Ressource Allocation

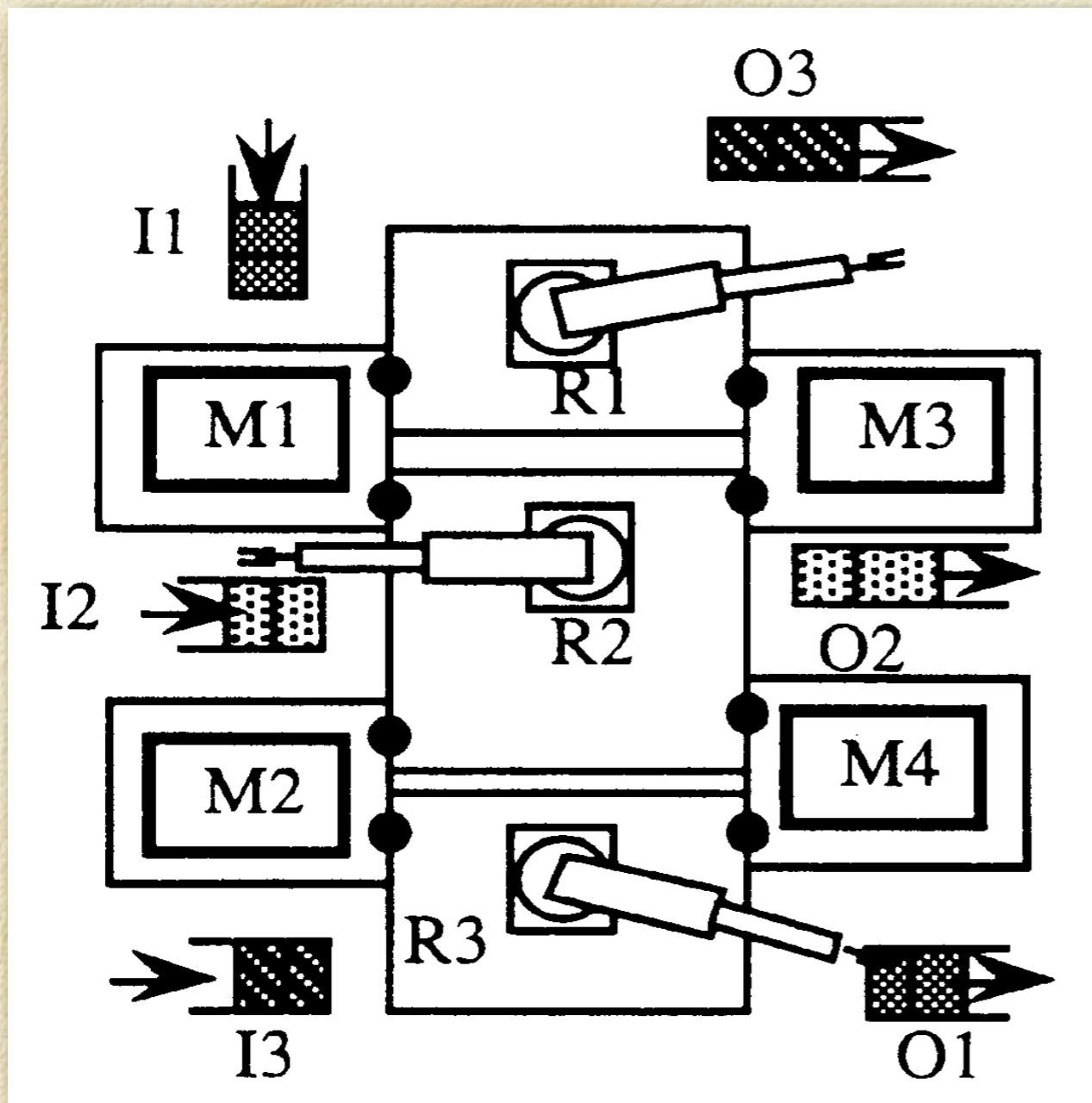
*bankers like resource
allocation
problems/algorithms*

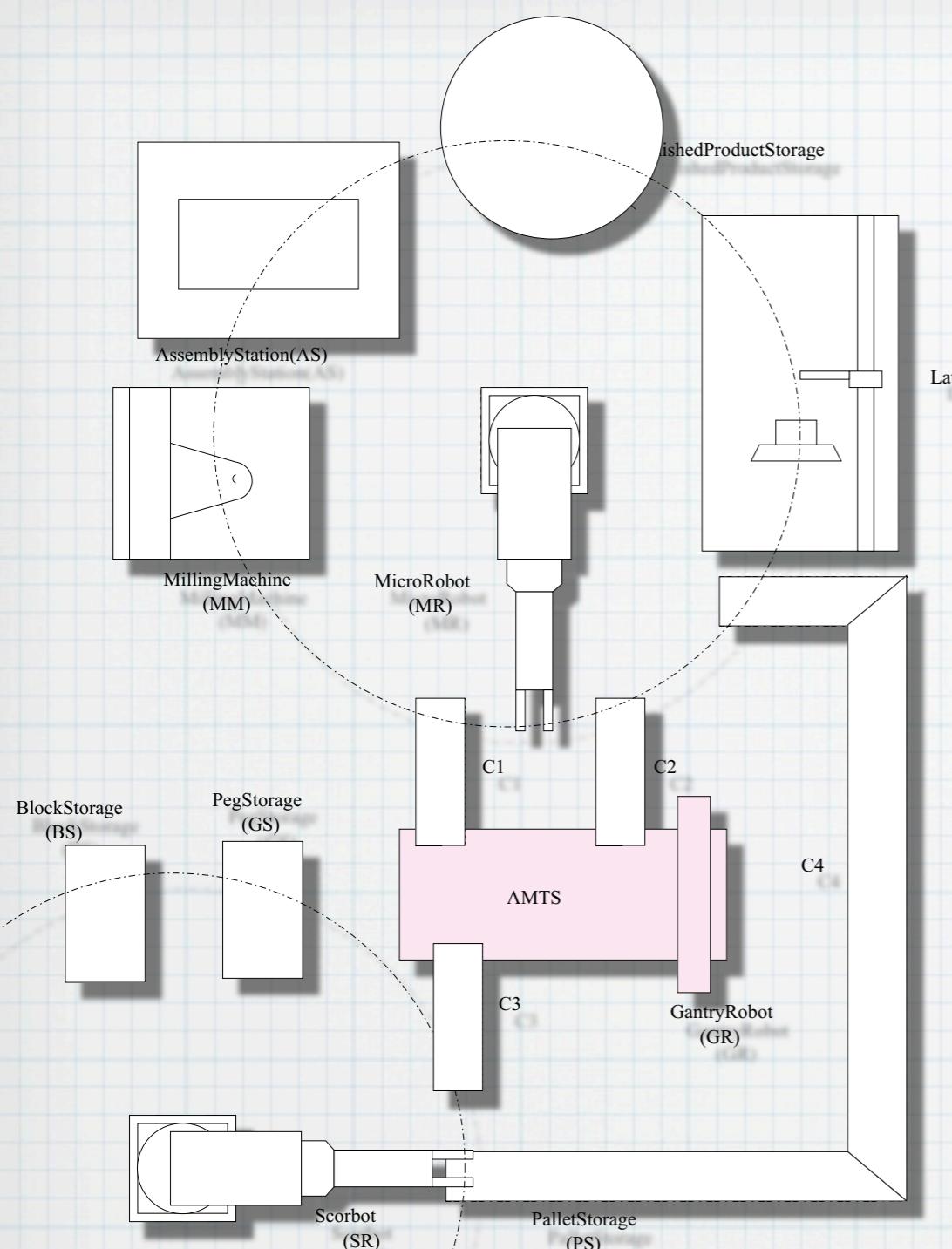
A Banker's solution for deadlock avoidance in FMS
with flexible routing and multi-resource states

J. Ezpeleta, F. Tricas, F.García-Vallés, J.M. Colom
Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior – Universidad de Zaragoza

María de Luna 3, 50015 Zaragoza (SPAIN)
Phone: 34 976 761955 Fax: 34 976 761861

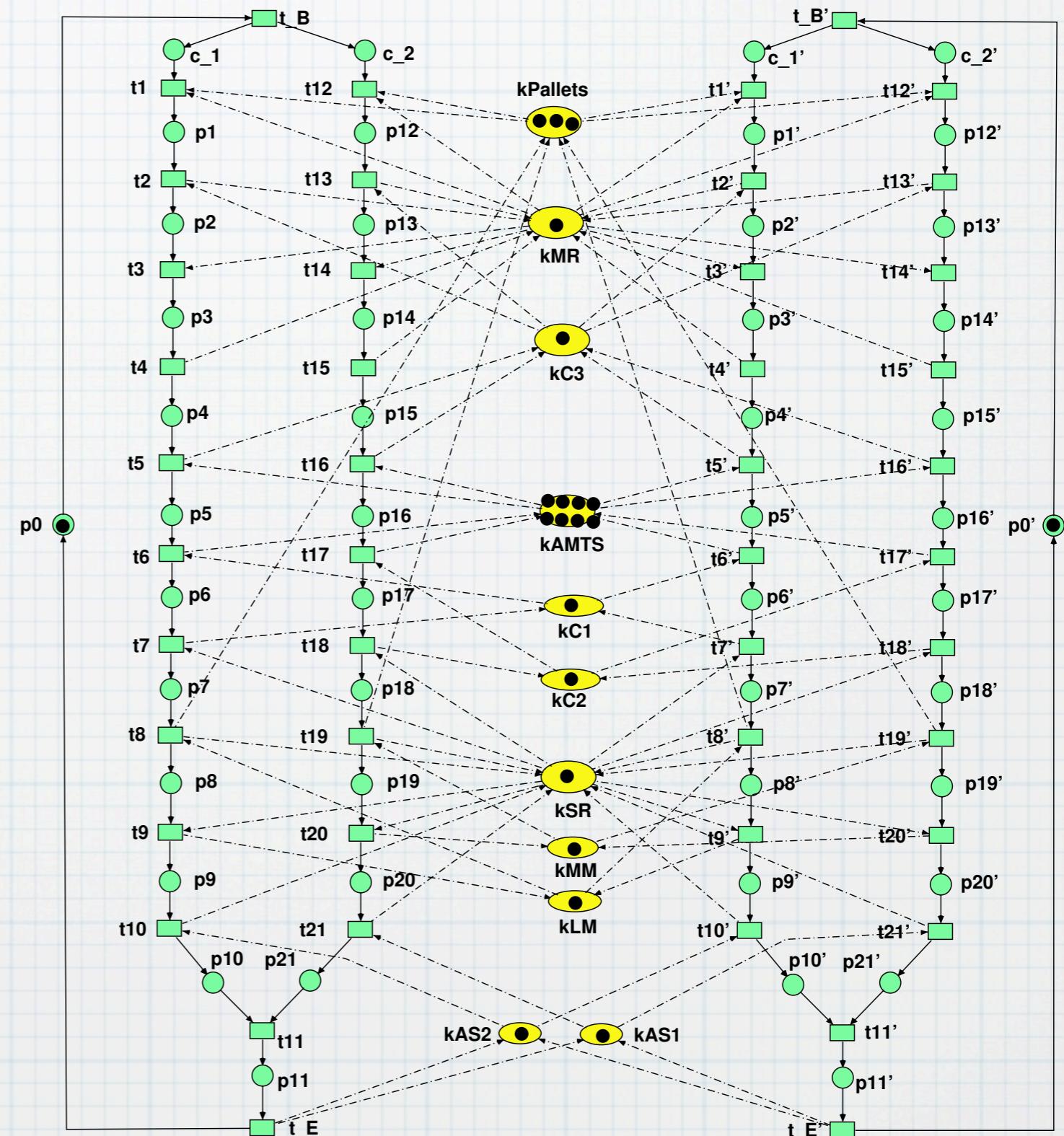
Corresponding Author: ezpeleta@posta.unizar.es
DRAFT PAPER





Bankiers-Problem

Bankers like resource allocation problems

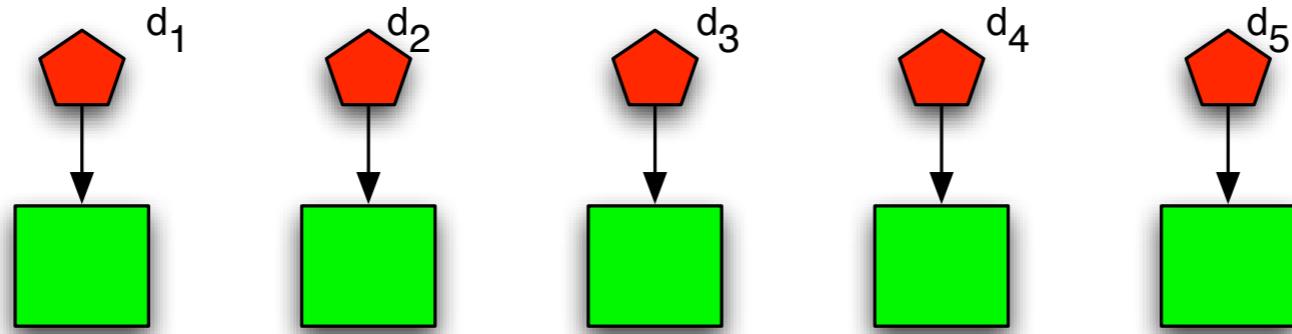


FGI 2

Daniel Moldt

Datenbankmanager

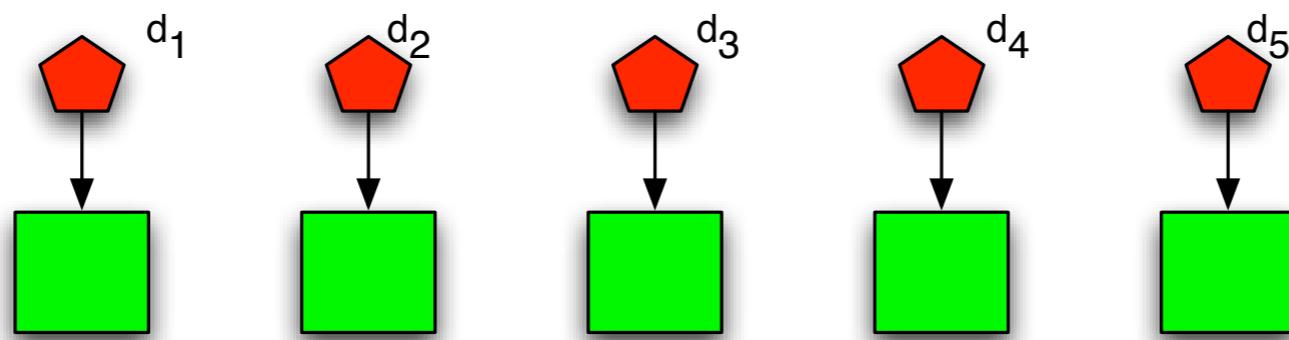
Der Datenbank-Manager



Das Beispiel der Datenbank-Manager⁵

Eine Menge von $n > 0$ Datenbanken soll von n Prozessen, genannt *Datenbank-Manager*, $DBM = \{d_1, d_2, \dots, d_n\}$ so verwaltet werden, dass sie immer den gleichen Inhalt haben. Um dies zu erreichen, sollen die Manager miteinander kommunizieren. Jeder Manager kann seine eigene Datenbank aktualisieren. Dabei muß er an jeden anderen Manager eine Nachricht senden, die diesen über die Aktualisierung informiert. Der Manager muß warten, bis alle anderen diese Nachricht erhalten, die Aktualisierung durchgeführt und eine entsprechende Rückmeldung zurückgesandt haben. Erst dann kehrt der Manager in den Zustand *inactive* zurück.

Der Datenbank-Manager



Dabei sollen weder die Datenbanken noch ihre Aktualisierung dargestellt werden, sondern nur der Nachrichtenaustausch.

Modell

Zustände der Manager : *inactive, waiting, performing*

Nachrichten : $MS = \{(s, r) | s, r \in DBM \wedge s \neq r\}$

Zustände der Nachrichtenpuffer : *unused, sent, received, acknowledged*

wechselseitiger Ausschluß : *exclusion*

Spezifikation für das gefärbte Netz von Abb. 4.22:

Farben : $DBM = \{d_1, d_2, \dots, d_n\}$

$MS = \{(s, r) | s, r \in DBM \wedge s \neq r\}$

$E = \{e\}$

Modell

Variablen : $Var = \{e, r, s\}$

$dom(e) = E, \quad dom(r) = dom(s) = DBM$

Funktionen :

$MINE : DBM \rightarrow Bag(MS)$

$MINE(s) := \sum_{r \neq s} (s, r)$

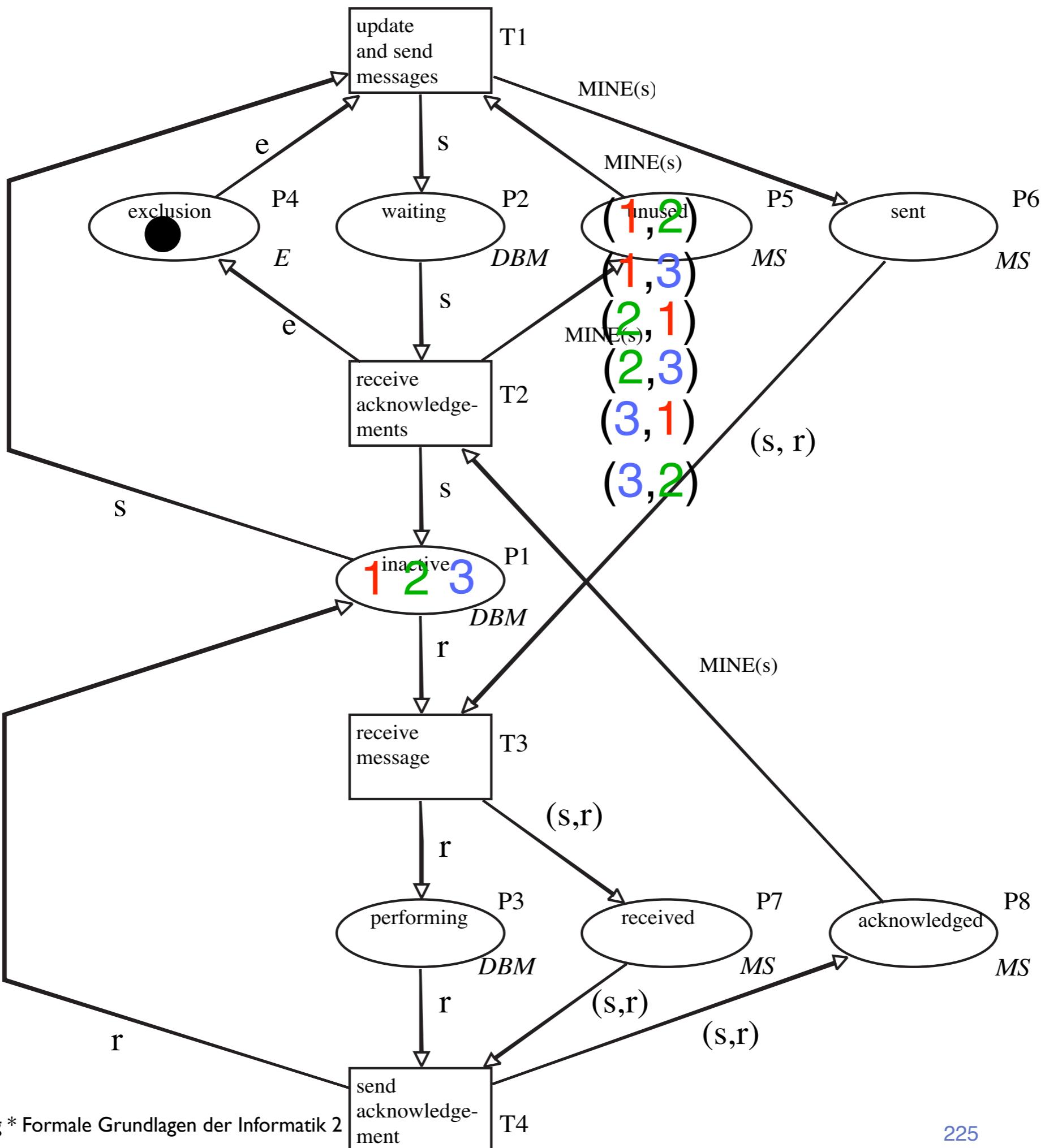
$REC : MS \rightarrow DBM$

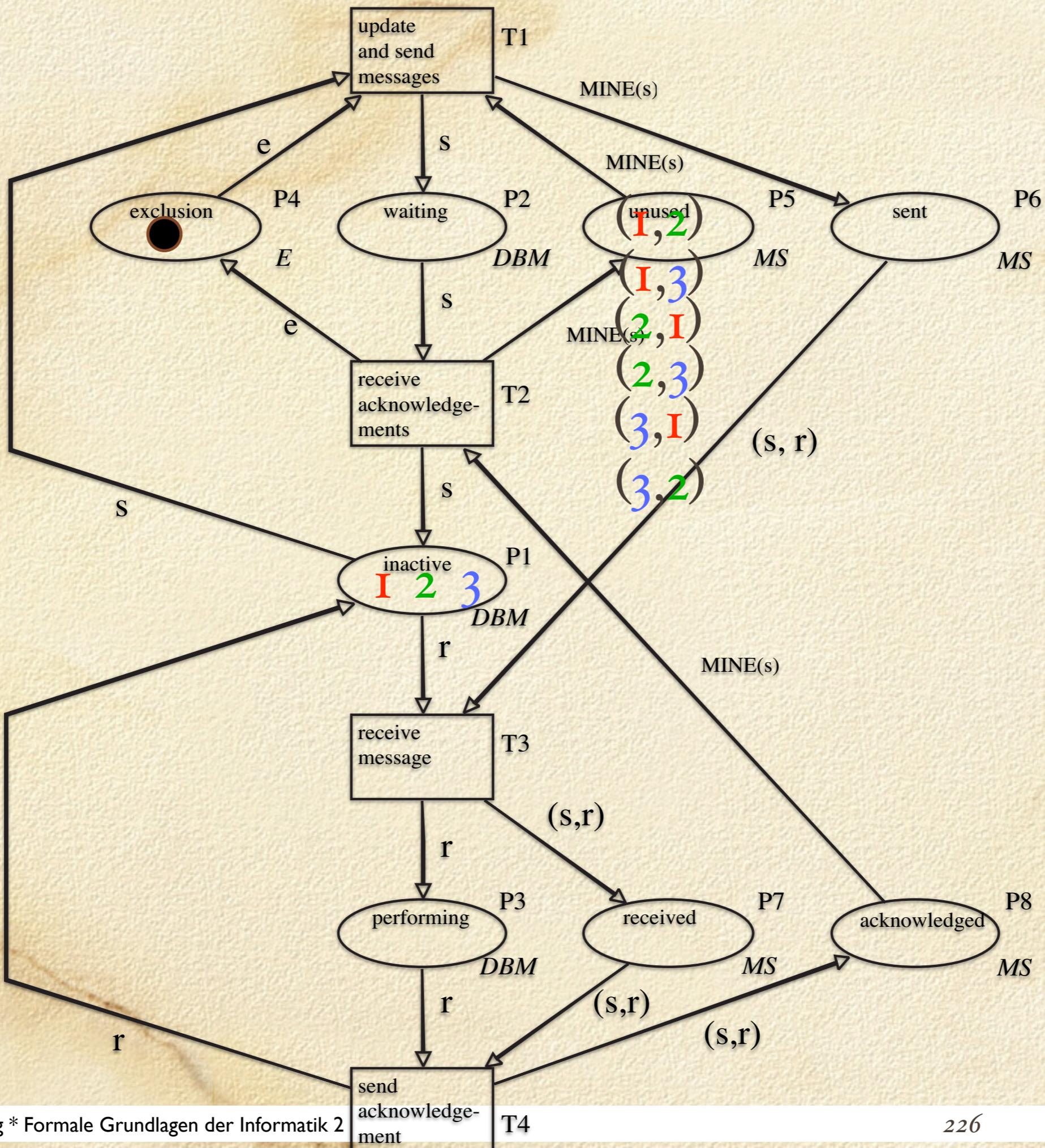
$REC((s, r)) := r$

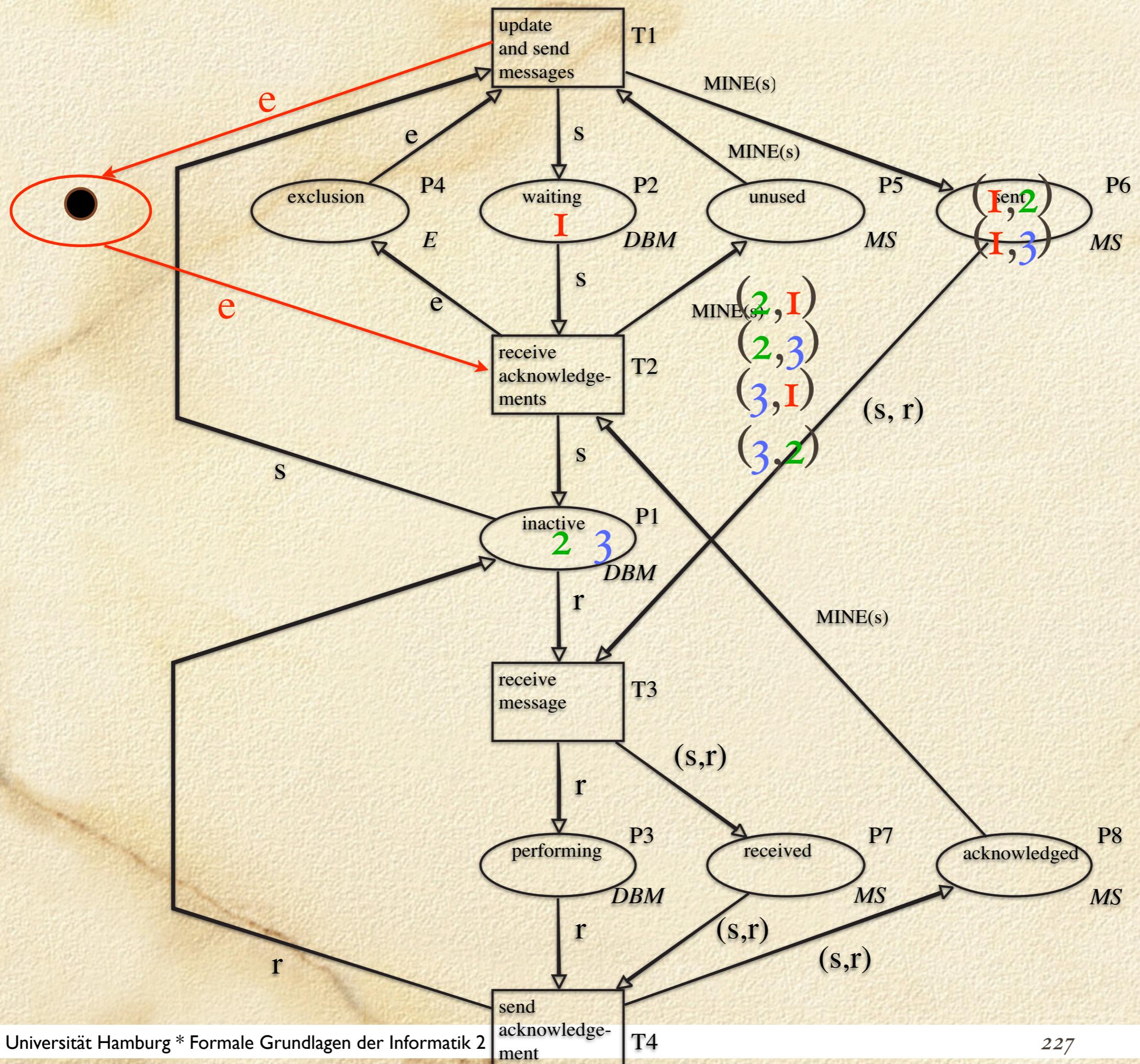
$ABS : DBM \rightarrow E$

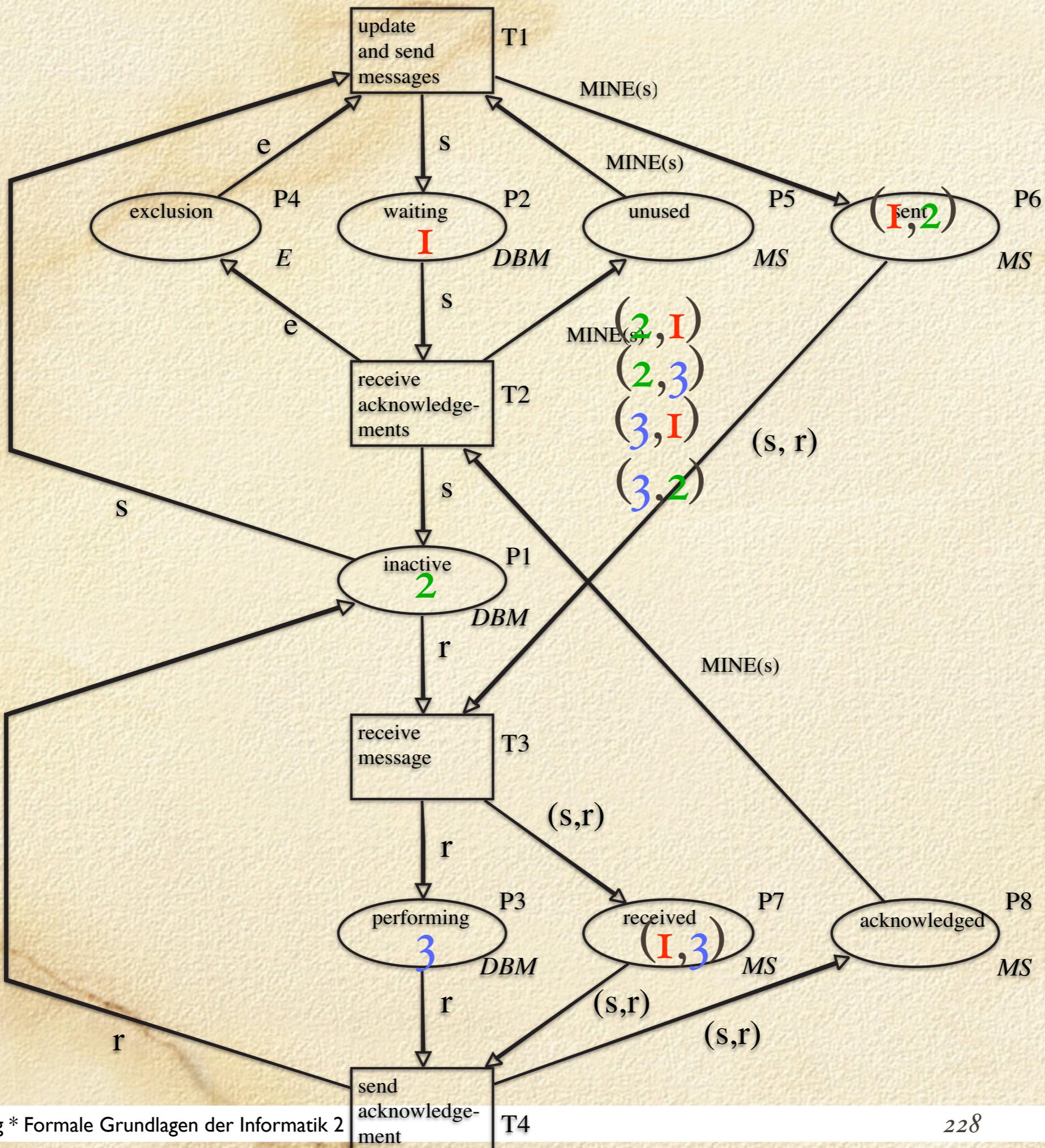
$ABS(s) := e$

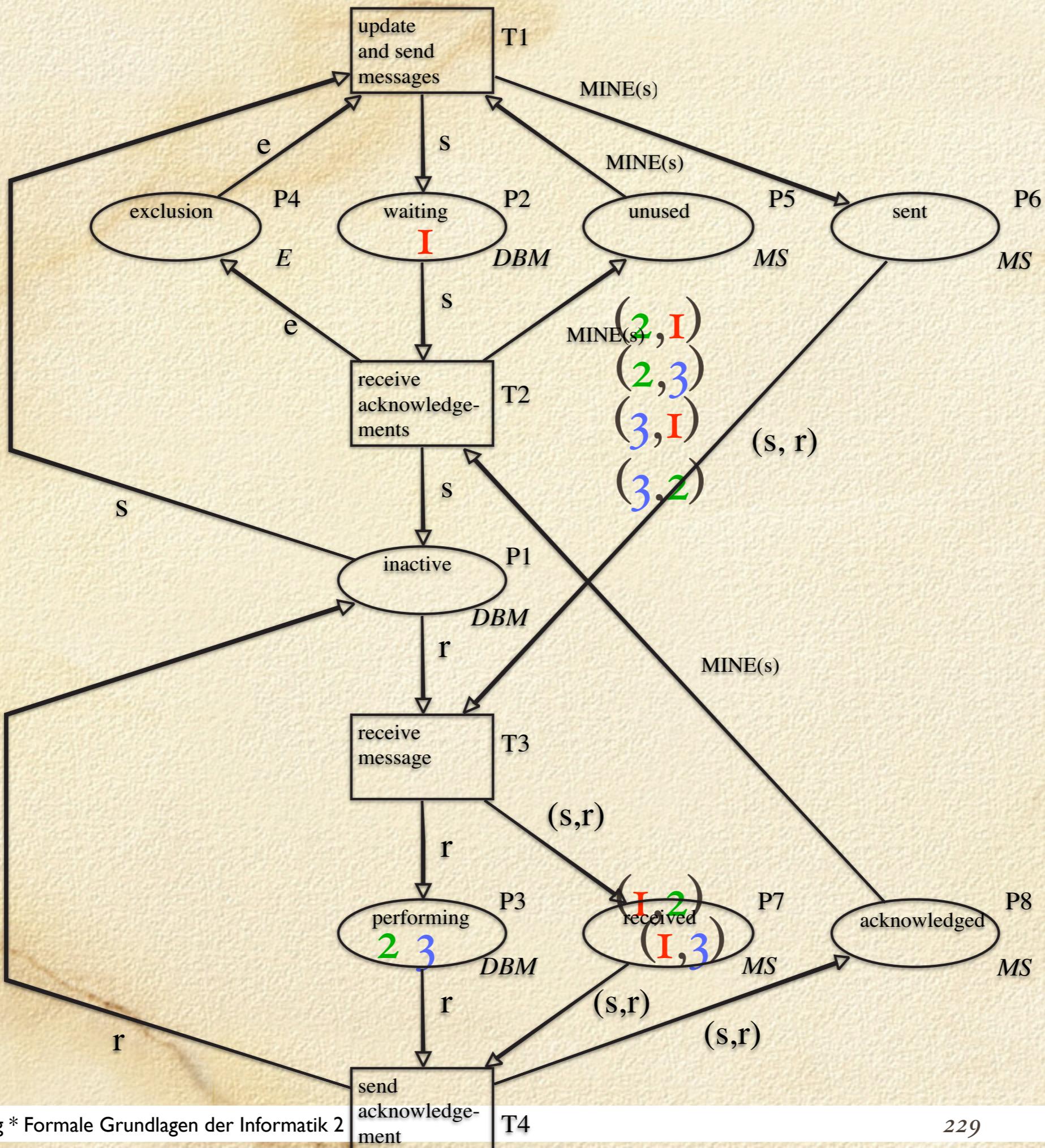
Anfangsmarkierung: $\mathbf{m}_0(p) := \begin{cases} DBM & \text{falls } p = \text{inactive} \\ MS & \text{falls } p = \text{unused} \\ \{e\} & \text{falls } p = \text{exclusion} \\ \emptyset & \text{sonst} \end{cases}$

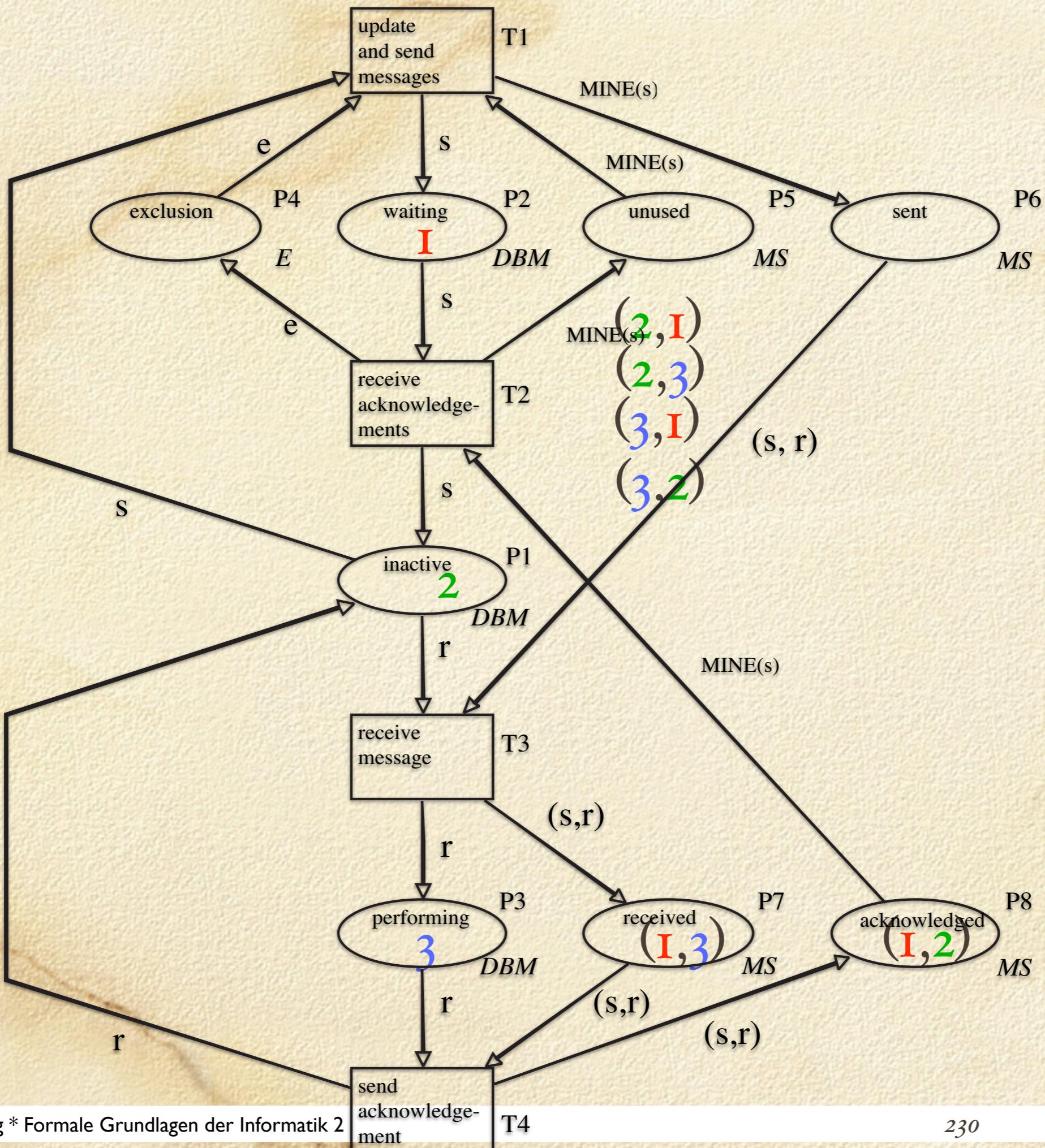


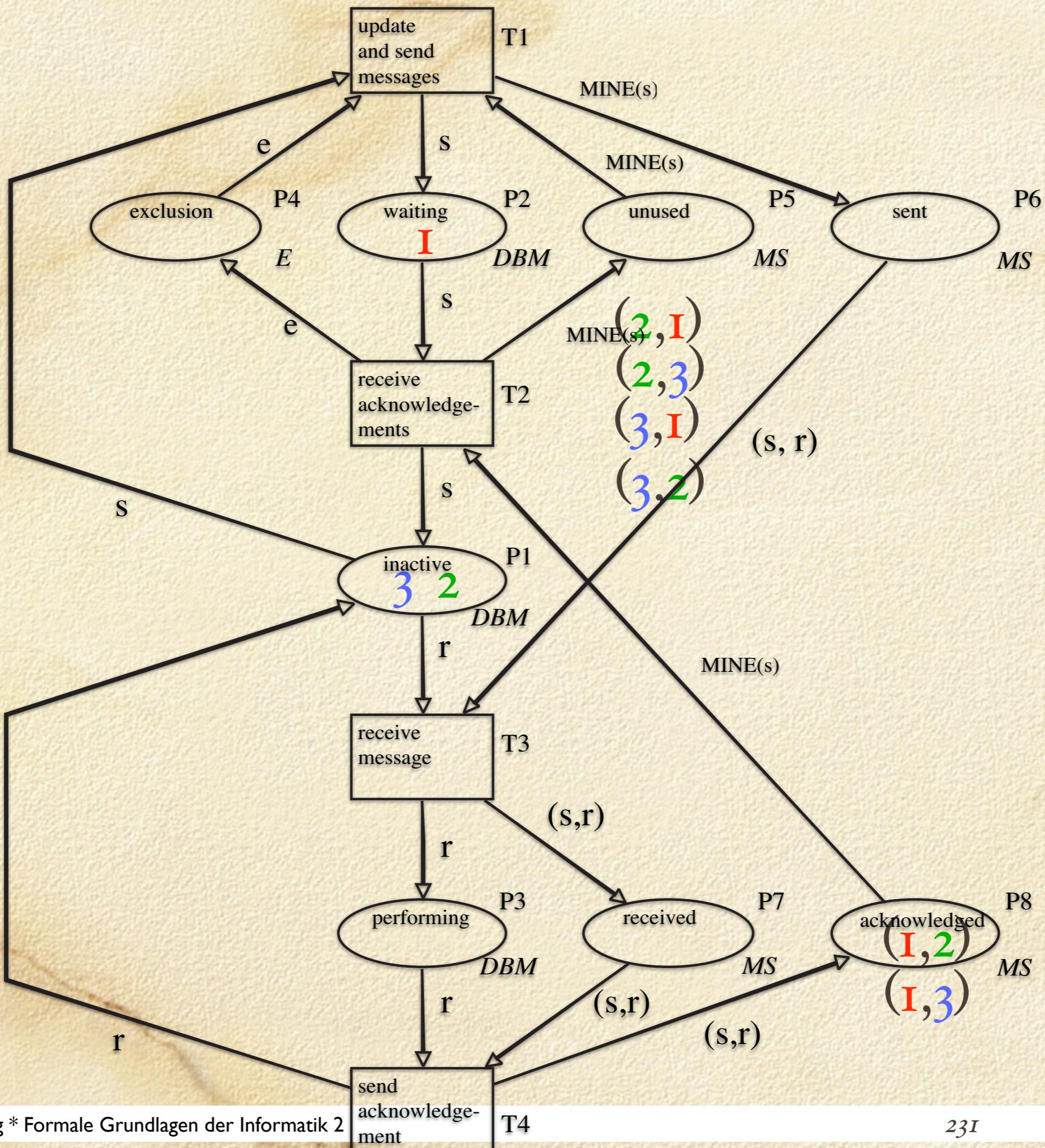


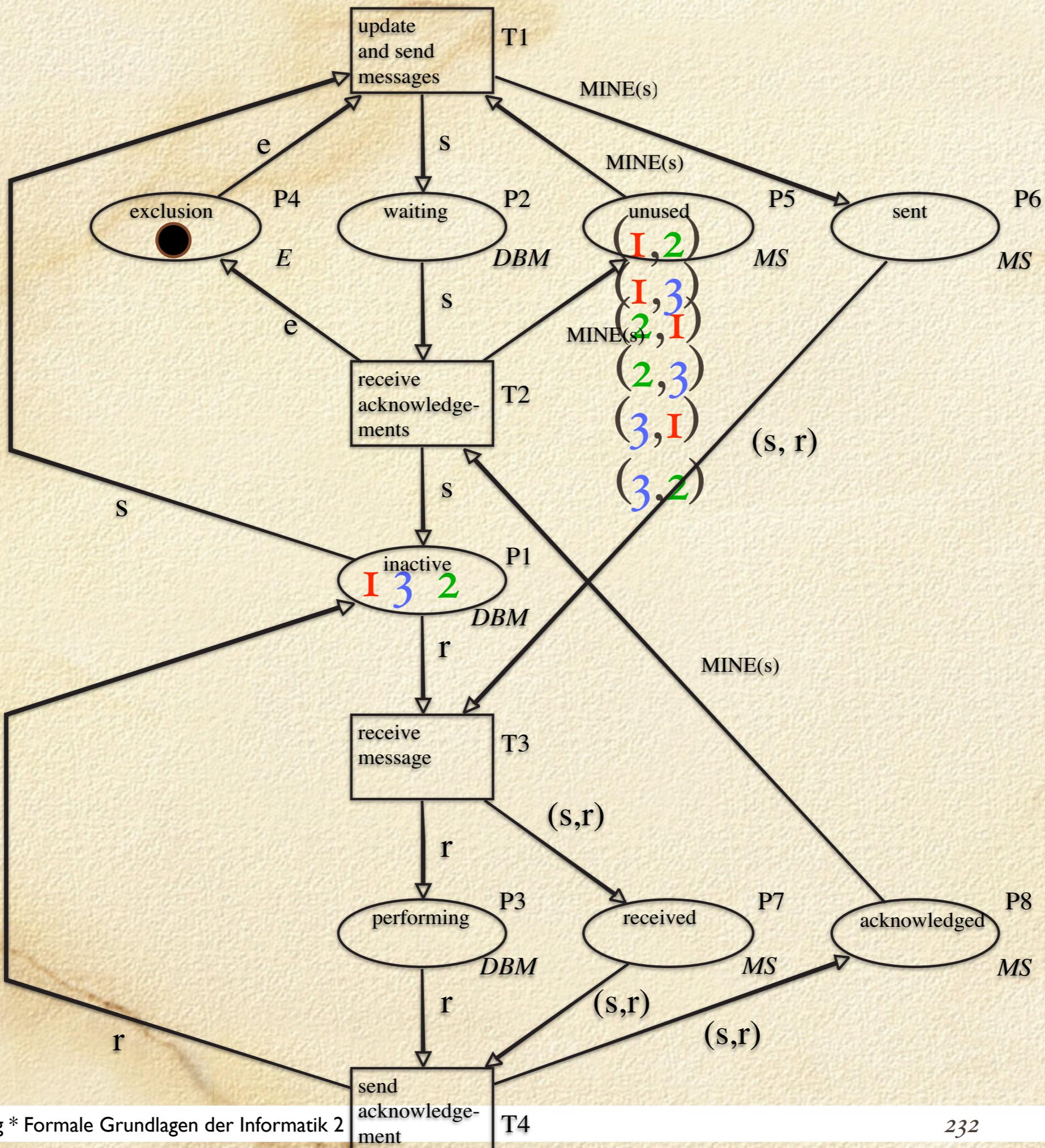




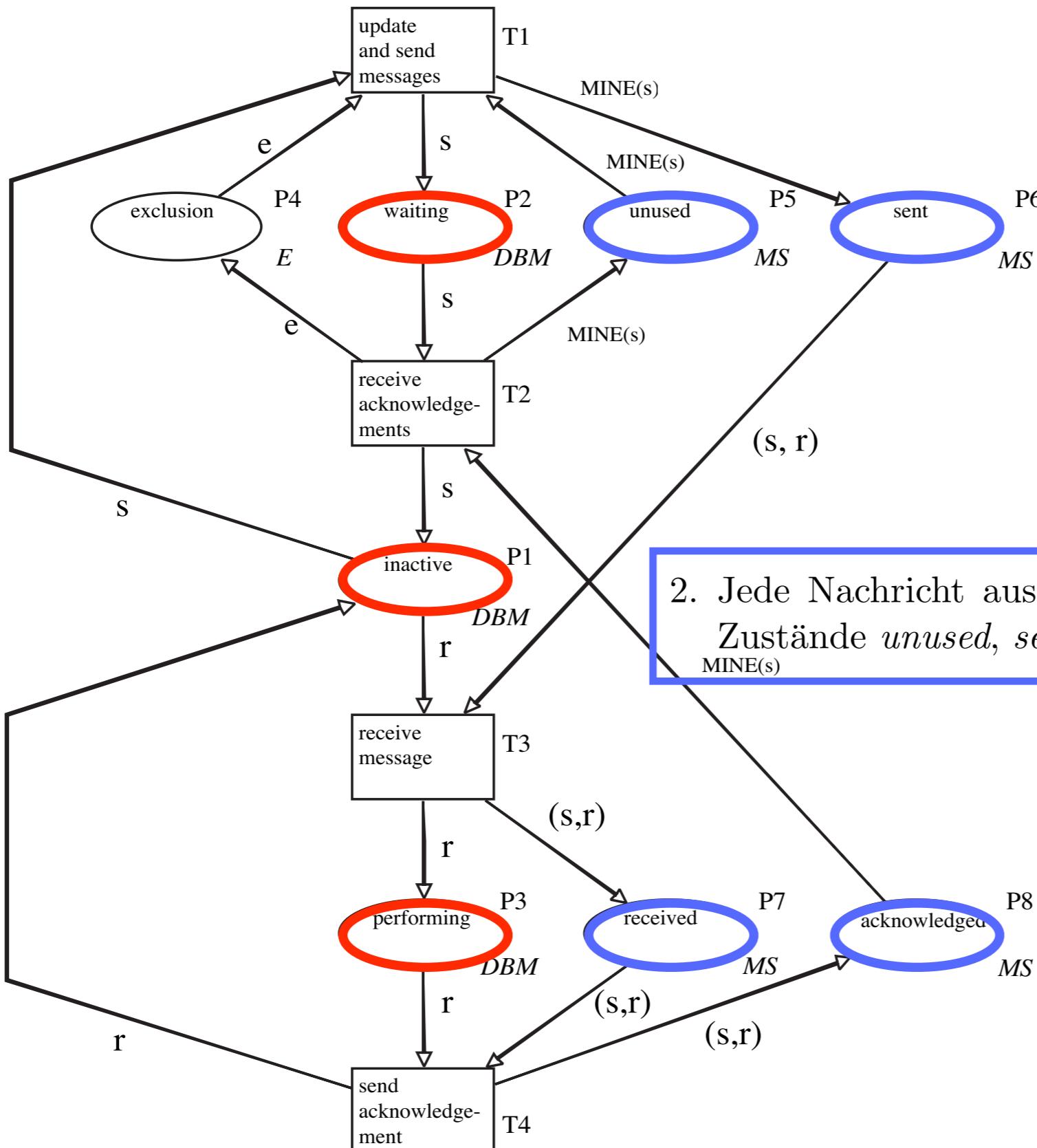






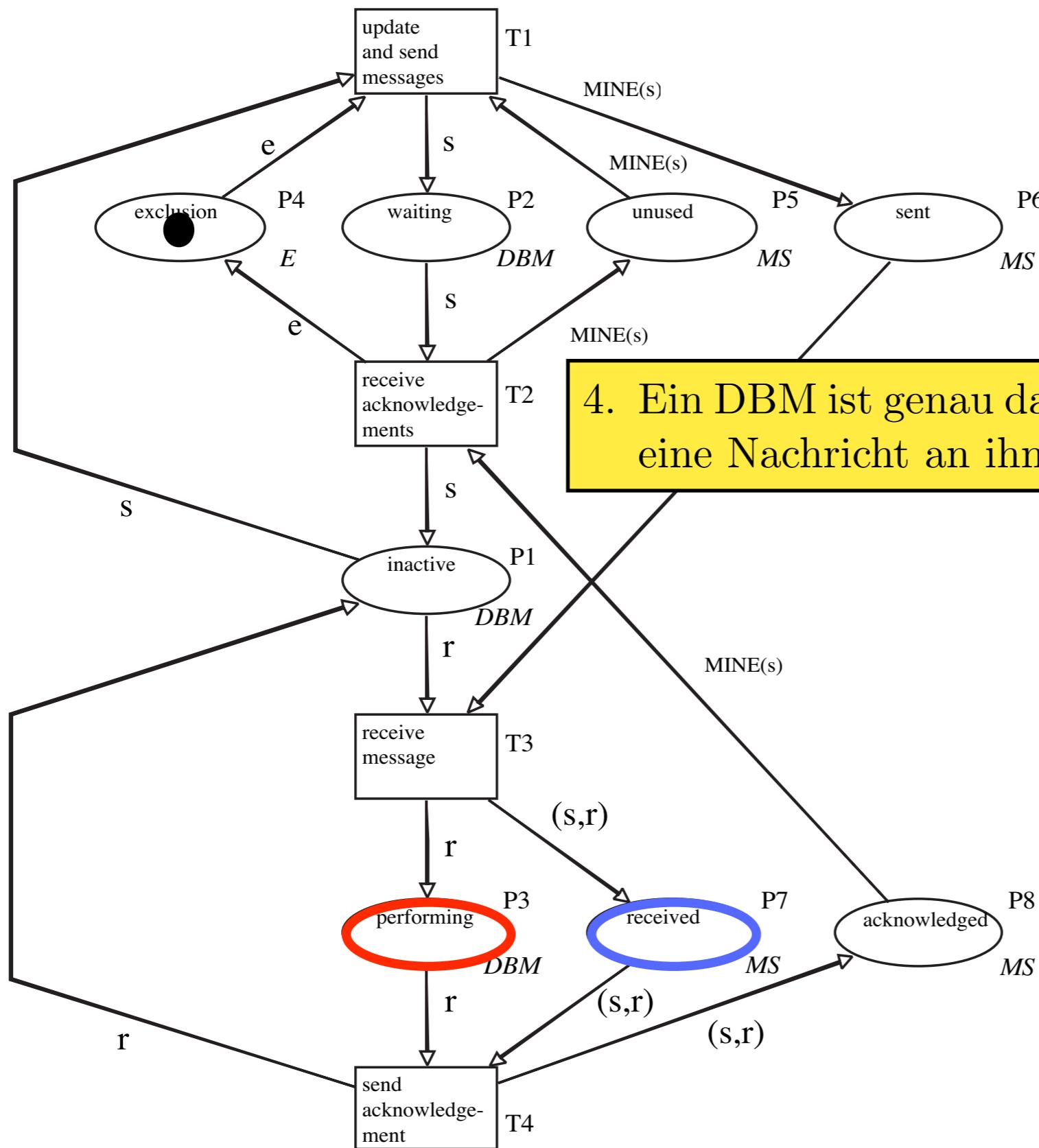


1. Jeder DBM ist in genau einem der drei Zustände *inactive*, *waiting* oder *performing*.

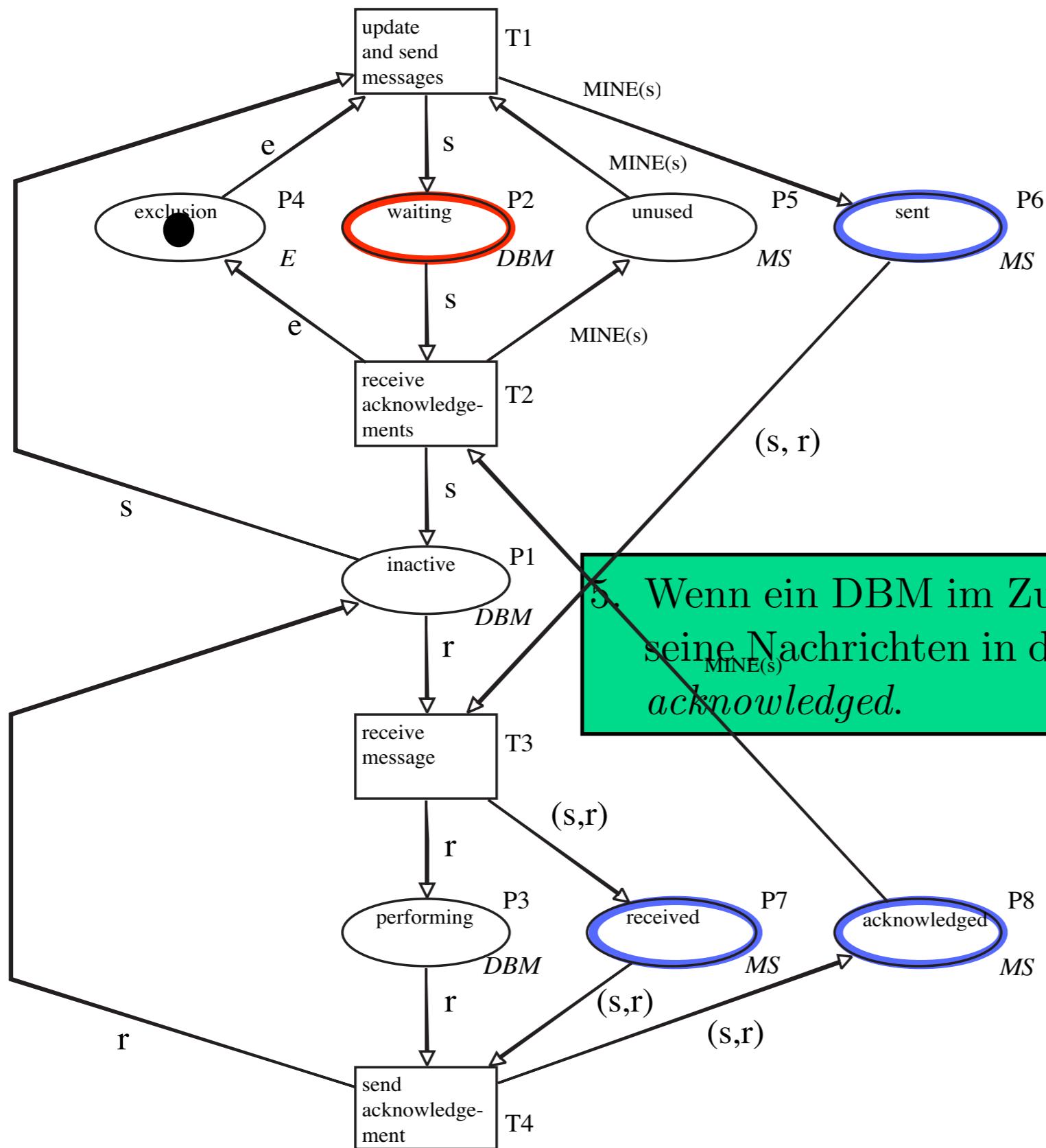


2. Jede Nachricht aus MS ist in genau einem der vier Zustände *unused*, *sent*, *received* oder *acknowledged*.

3. Höchstens ein DBM wartet.



4. Ein DBM ist genau dann im Zustand *performing*, wenn eine Nachricht an ihn im Zustand *received* ist.



5. Wenn ein DBM im Zustand *waiting* ist, dann sind alle seine Nachrichten in den Zuständen *sent*, *received* oder *acknowledged*.

Invarianten

Satz 4.18 Für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ des DBM-Netzes \mathcal{N} von Abbildung 4.21 gilt:

1. Jeder DBM ist in genau einem der drei Zustände *inactive*, *waiting* oder *performing*.

$$1. \mathbf{m}(\text{inactive}) + \mathbf{m}(\text{waiting}) + \mathbf{m}(\text{performing}) = DBM$$

Multimengen-
vereinigung

2. Jede Nachricht aus MS ist in genau einem der vier Zustände *unused*, *sent*, *received* oder *acknowledged*.

$$\mathbf{m}(\text{unused}) + \mathbf{m}(\text{sent}) + \mathbf{m}(\text{received}) + \mathbf{m}(\text{acknowledged}) = MS$$

3. Höchstens ein DBM wartet.

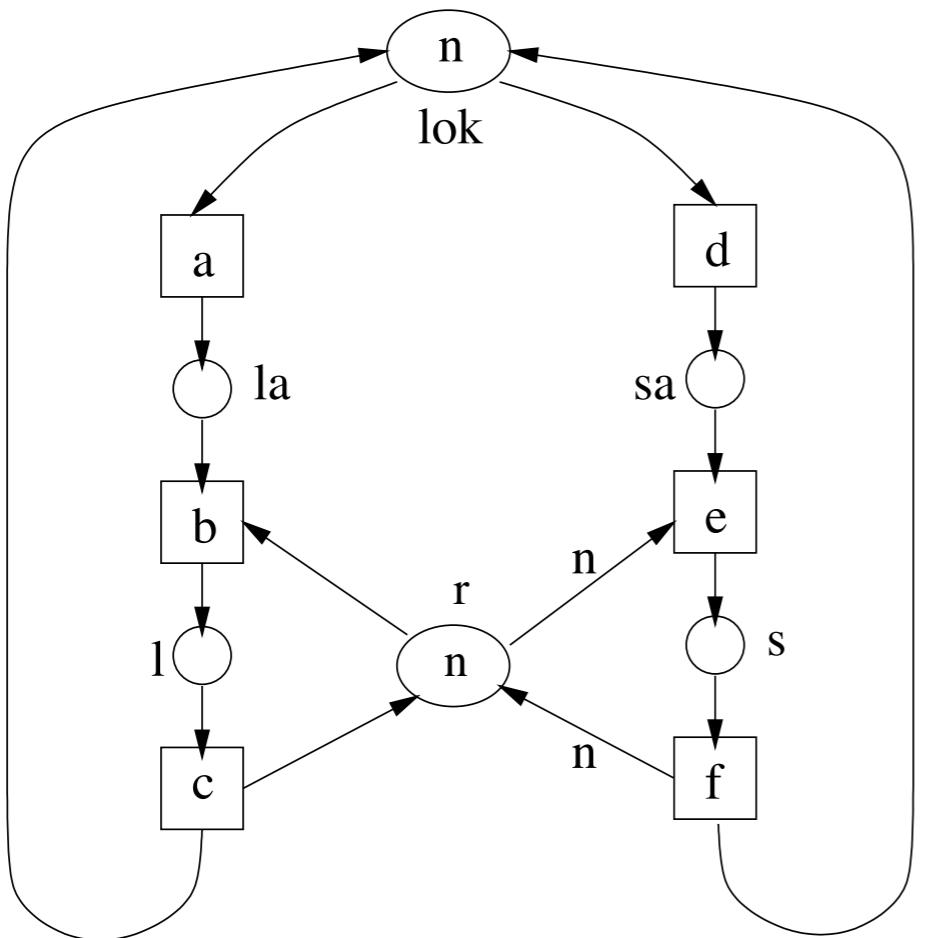
$$3. ABS(\mathbf{m}(\text{waiting})) + \mathbf{m}(\text{exclusion}) = \{e\}$$

Invarianten

Satz 4.18 Für alle erreichbaren Markierungen $\mathbf{m} \in \mathbf{R}(\mathcal{N})$ des DBM-Netzes \mathcal{N} von Abbildung 4.21 gilt:

4. Ein DBM ist genau dann im Zustand *performing*, wenn eine Nachricht an ihn im Zustand *received* ist.
4. $\mathbf{m}(\textit{performing}) = REC(\mathbf{m}(\textit{received}))$
5. Wenn ein DBM im Zustand *waiting* ist, dann sind alle seine Nachrichten in den Zuständen *sent*, *received* oder *acknowledged*.
5. $MIN E(\mathbf{m}(\textit{waiting})) = \mathbf{m}(\textit{sent}) + \mathbf{m}(\textit{received}) + \mathbf{m}(\textit{acknowledged})$

Invariantenkalkül: P/T-Netze



- $i_1 : lok + la + sa + l + s = n$

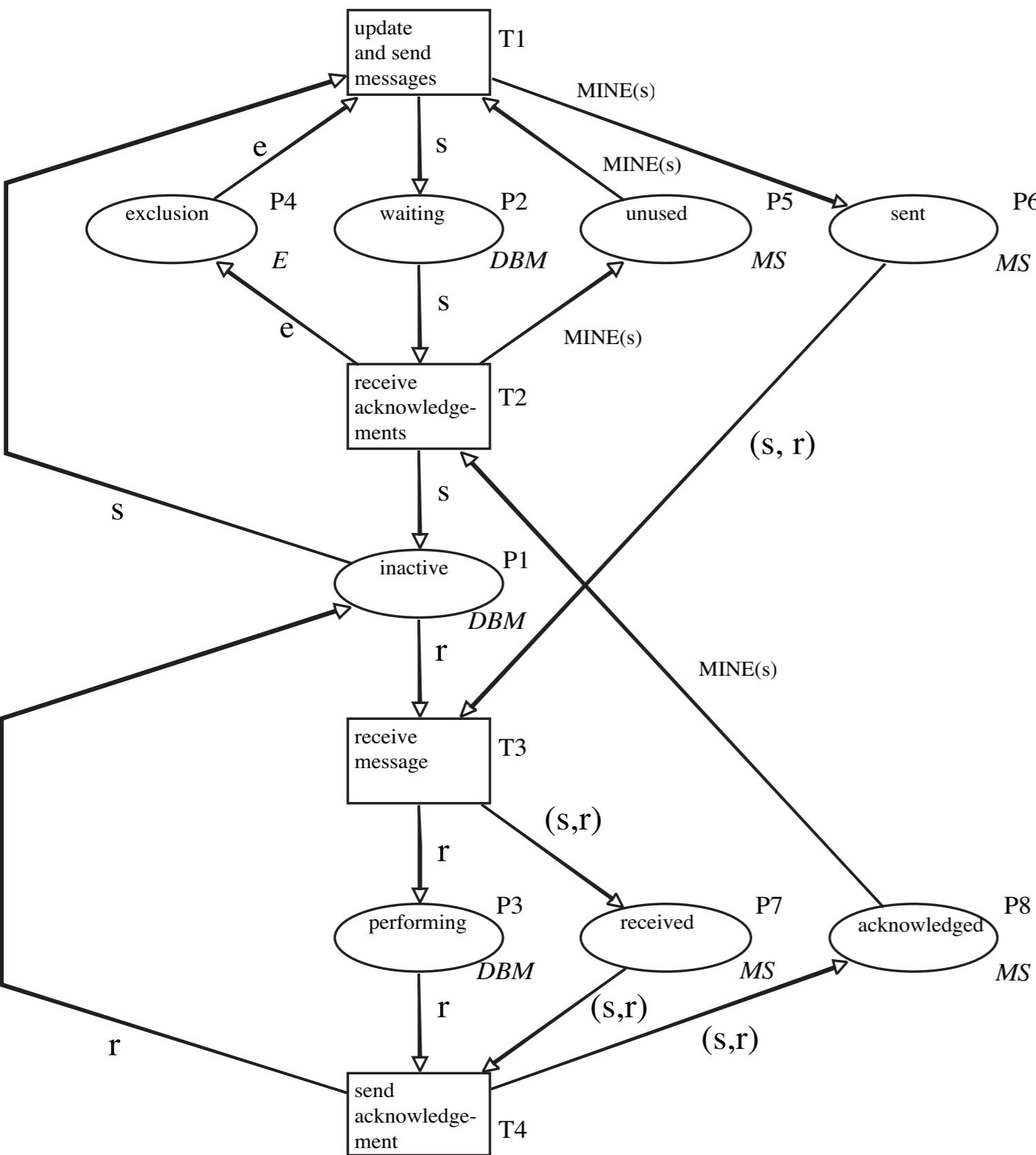
- $i_2 : l + r + n \cdot s = n$

(lok, la, \dots stehen hier abkürzend für $\mathbf{m}(lok), \mathbf{m}(la), \dots$)

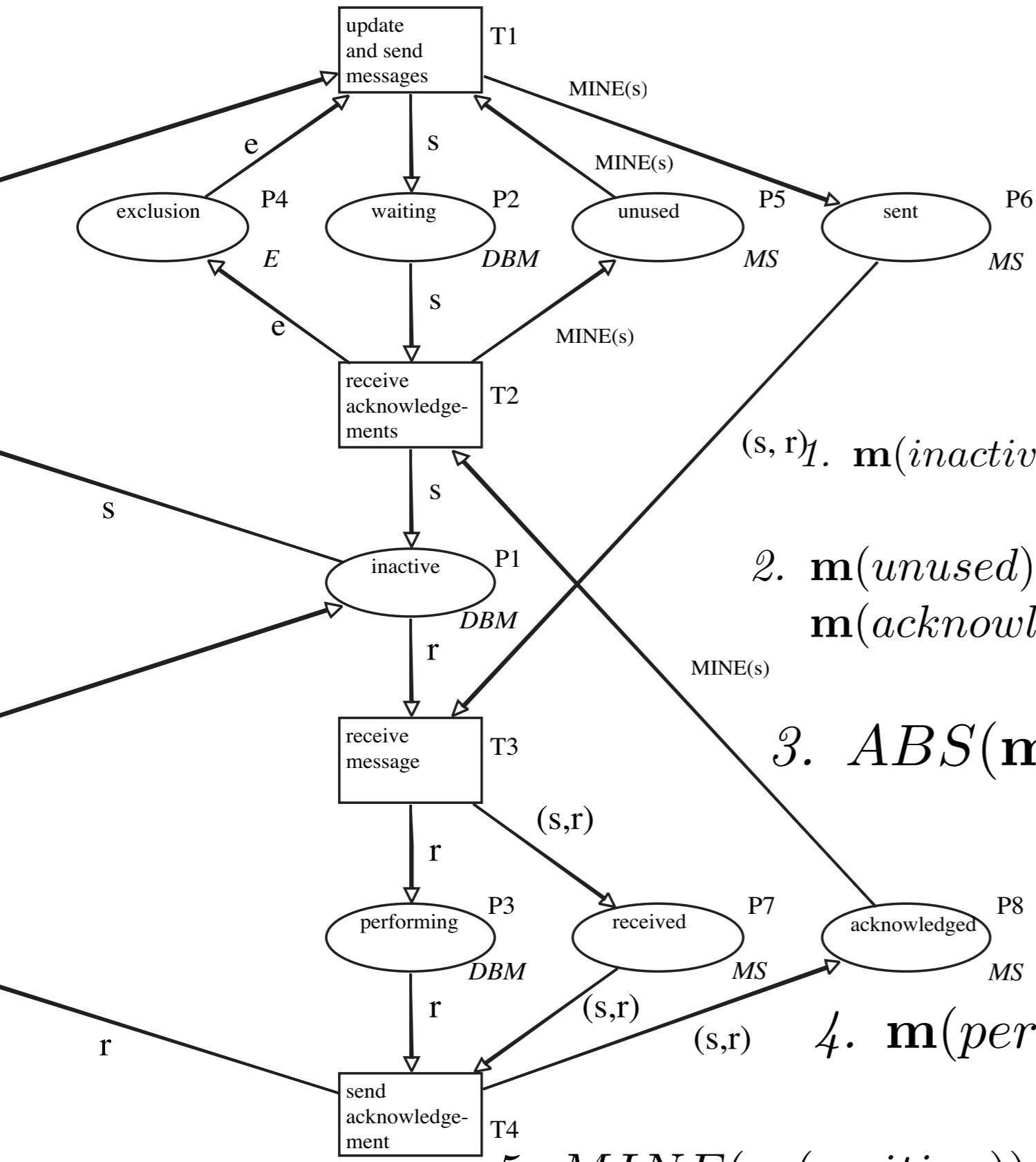
Abbildung 4.1: Leser/Schreiber-Problem für n Aufträge

\mathcal{N}	a	b	c	d	e	f		i_1		i_2
lok	-1	0	1	-1	0	1		1		0
la	1	-1	0	0	0	0		1		0
sa	0	0	0	1	-1	0		1		0
l	0	1	-1	0	0	0		1		1
s	0	0	0	0	1	-1		1		n
r	0	-1	1	0	$-n$	n		0		1

Invariantenkalkül: CPN



DATABASE SYSTEM		T1	T2	T3	T4	m ₀
		DBM	DBM	MS	MS	
inactive	DBM	-ID	ID	-REC	REC	ΣDBM
waiting	DBM	ID	-ID			
performing	DBM			REC	-REC	
exclusion	E	-ABS	ABS			E
unused	MS	-MINE	MINE			ΣMS
sent	MS	MINE		-ID		
received	MS		ID	-ID		
acknowledged	MS		-MINE	ID		



$$1. \mathbf{m}(inactive) + \mathbf{m}(waiting) + \mathbf{m}(performing) = DBM$$

$$2. \mathbf{m}(unused) + \mathbf{m}(sent) + \mathbf{m}(received) + \mathbf{m}(acknowledged) = MB$$

$$3. ABS(\mathbf{m}(waiting)) + \mathbf{m}(exclusion) = \{e\}$$

$$4. \mathbf{m}(performing) = REC(\mathbf{m}(received))$$

$$5. MINE(\mathbf{m}(waiting)) = \mathbf{m}(sent) + \mathbf{m}(received) + \mathbf{m}(acknowledged)$$

$$1. \mathbf{m}(inactive) + \mathbf{m}(waiting) + \mathbf{m}(performing) = DBM$$

DATABASE SYSTEM		T1	T2	T3	T4	m_0	w1
		DBM	DBM	MB	MB	DBM	
inactive	DBM	-ID	ID	-REC	REC	ΣDBM	
waiting	DBM	ID	-ID				
performing	DBM			REC	-REC		
exclusion	E	-ABS	ABS			e	
unused	MB	-MINE	MINE			ΣMB	
sent	MB	MINE		-ID			
received	MB			ID	-ID		
acknowledged	MB		-MINE		ID		

DATABASE SYSTEM		<i>Invariants</i>								
		T1	T2	T3	T4	\mathbf{m}_0	w1	w2	w3	w4
		DBM	DBM	MB	MB		DBM	MB	E	DBM
inactive	DBM	-ID	ID	-REC	REC	ΣDBM	ID			
waiting	DBM	ID	-ID				ID	ABS		MINE
performing	DBM			REC	-REC		ID		ID	
exclusion	E	-ABS	ABS			e		ID		
unused	MB	-MINE	MINE			ΣMB		ID		
sent	MB	MINE		-ID			ID		-ID	
received	MB			ID	-ID		ID	-REC	-ID	
acknowledged	MB		-MINE		ID		ID		-ID	

2. $\mathbf{m}(\text{unused}) + \mathbf{m}(\text{sent}) + \mathbf{m}(\text{received}) + \mathbf{m}(\text{acknowledged}) = MB$
3. $ABS(\mathbf{m}(\text{waiting})) + \mathbf{m}(\text{exclusion}) = \{e\}$
4. $\mathbf{m}(\text{performing}) = REC(\mathbf{m}(\text{received}))$
5. $MINE(\mathbf{m}(\text{waiting})) = \mathbf{m}(\text{sent}) + \mathbf{m}(\text{received}) + \mathbf{m}(\text{acknowledged})$

DATABASE SYSTEM		<i>Invariants</i>									
		T1	T2	T3	T4	\mathbf{m}_0	w1	w2	w3	w4	w5
		DBM	DBM	MB	MB		DBM	MB	E	DBM	MB
inactive	DBM	-ID	ID	-REC	REC	ΣDBM	ID				
waiting	DBM	ID	-ID				ID	ABS			MINE
performing	DBM			REC	-REC		ID		ID		
exclusion	E	-ABS	ABS			e		ID			
unused	MB	-MINE	MINE			ΣMB		ID			
sent	MB	MINE		-ID			ID			-ID	
received	MB			ID	-ID				-REC	-ID	
acknowledged	MB		-MINE		ID		ID			-ID	

$$2. \mathbf{m}(\text{unused}) + \mathbf{m}(\text{sent}) + \mathbf{m}(\text{received}) + \mathbf{m}(\text{acknowledged}) = MB$$

$$3. ABS(\mathbf{m}(\text{waiting})) + \mathbf{m}(\text{exclusion}) = \{e\}$$

$$4. \mathbf{m}(\text{performing}) = REC(\mathbf{m}(\text{received}))$$

$$5. MINE(\mathbf{m}(\text{waiting})) = \mathbf{m}(\text{sent}) + \mathbf{m}(\text{received}) + \mathbf{m}(\text{acknowledged})$$

FGI 2

Daniel Moldt

Alternierbitprotokoll

Alternierbitprotokoll in RENEW

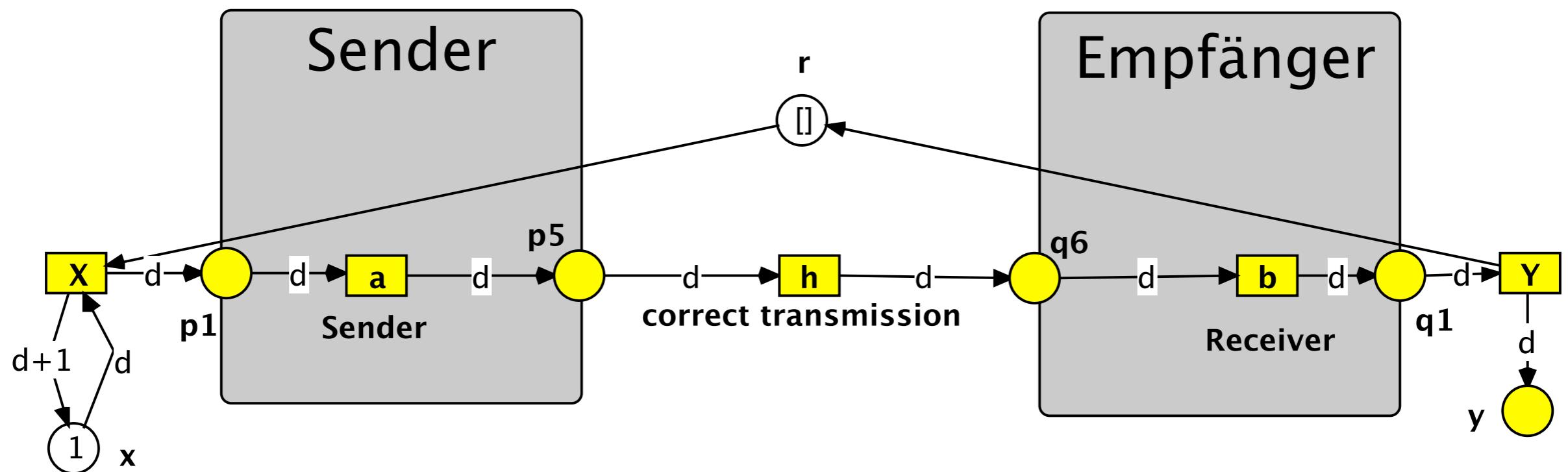


Abbildung 2.56: Spezifikation abp-1 des Alternierbitprotokolls

Alternierbitprotokoll in RENEW

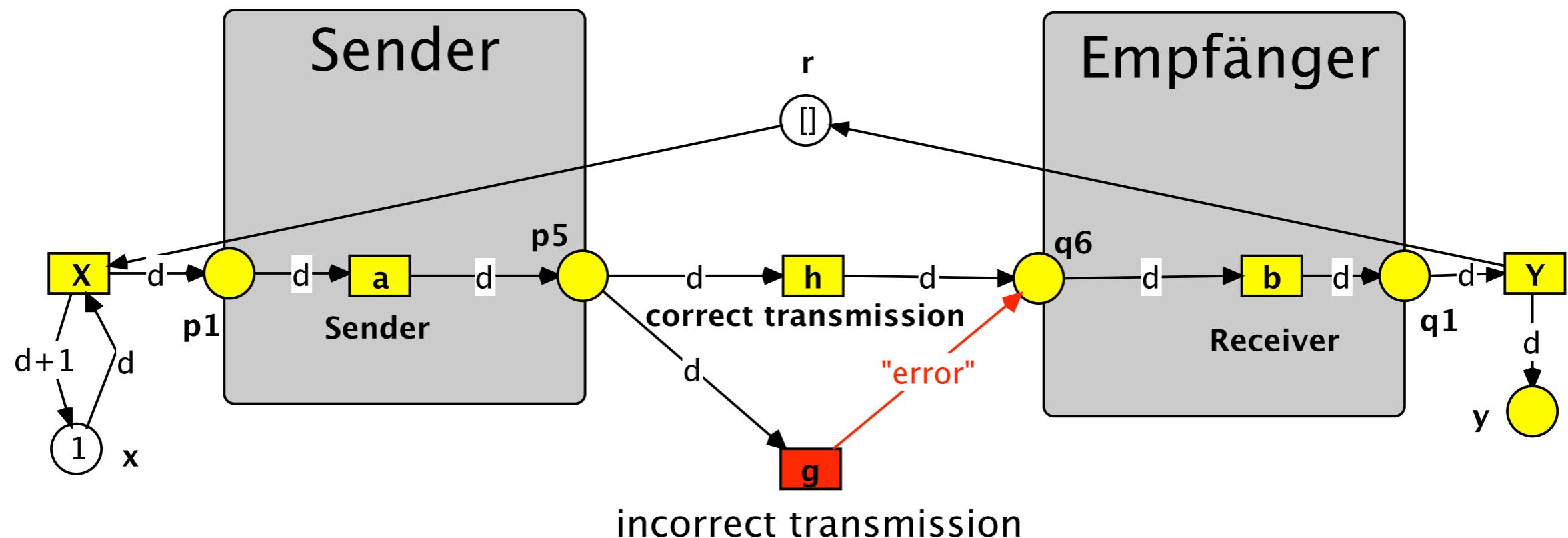


Abbildung 2.57: Übermittlung mit Fehlern: Netz abp-2

Alternierbitprotokoll in RENEW

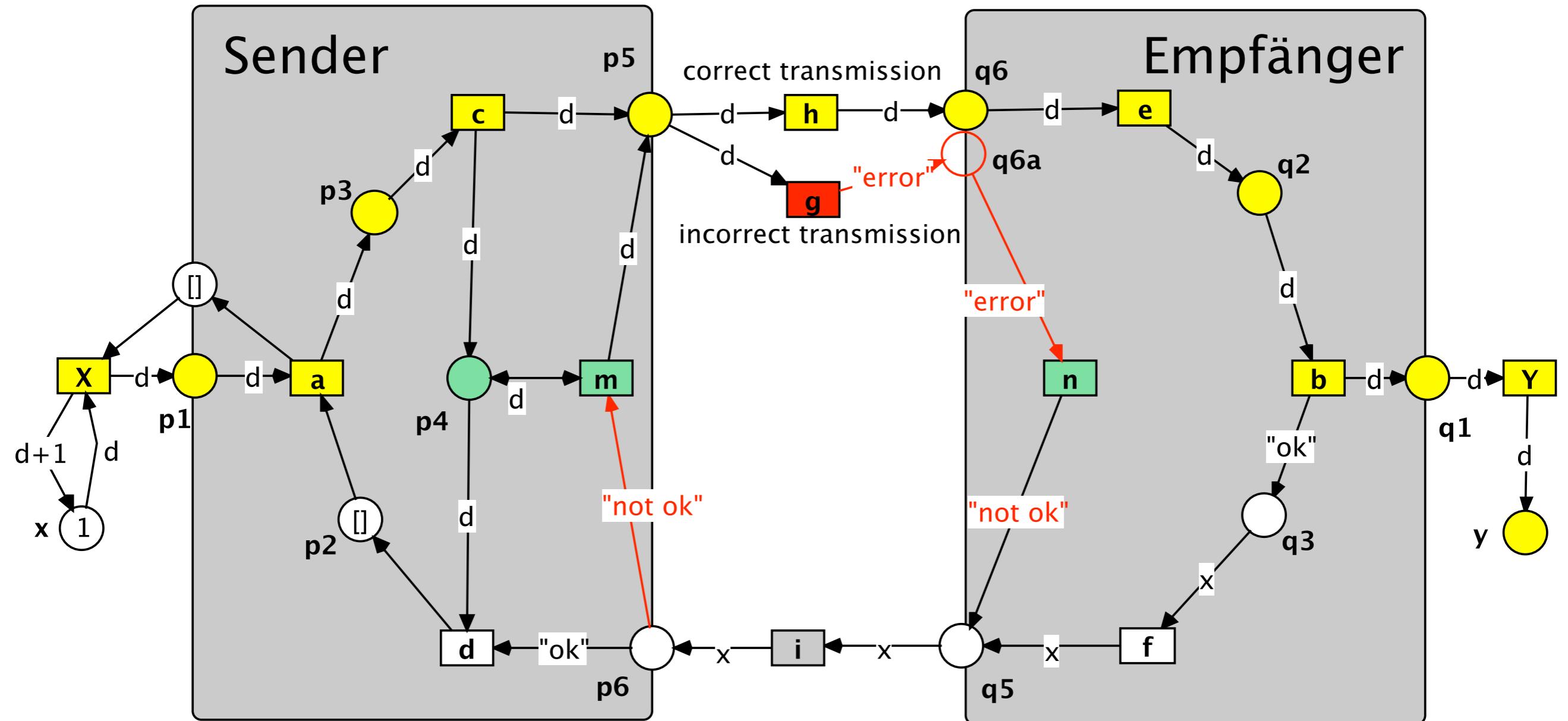


Abbildung 2.58: Übermittlung mit Quittung: Netz abp-3

Alternierbitprotokoll in RENEW

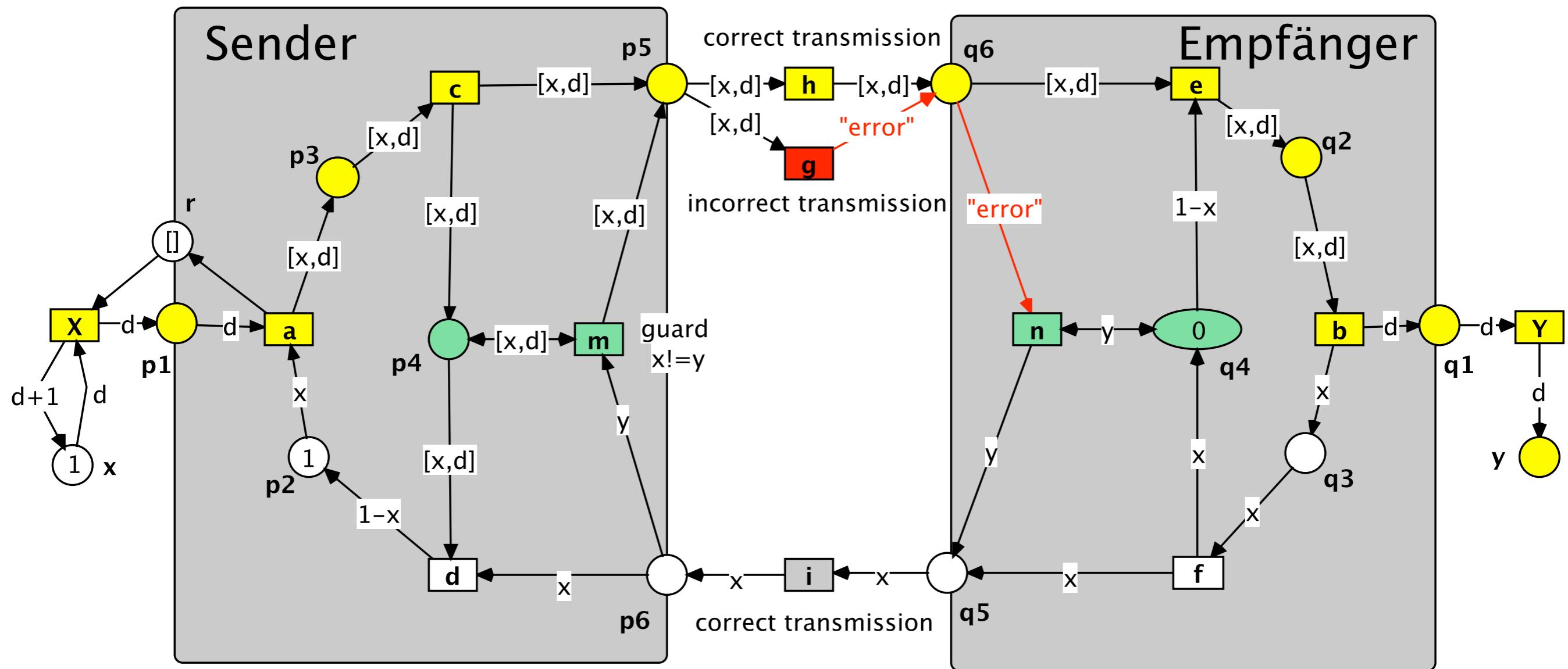


Abbildung 2.59: Übermittlung mit Quittung durch Alternierbit: Netz abp-4

Alternierbitprotokoll in RENEW

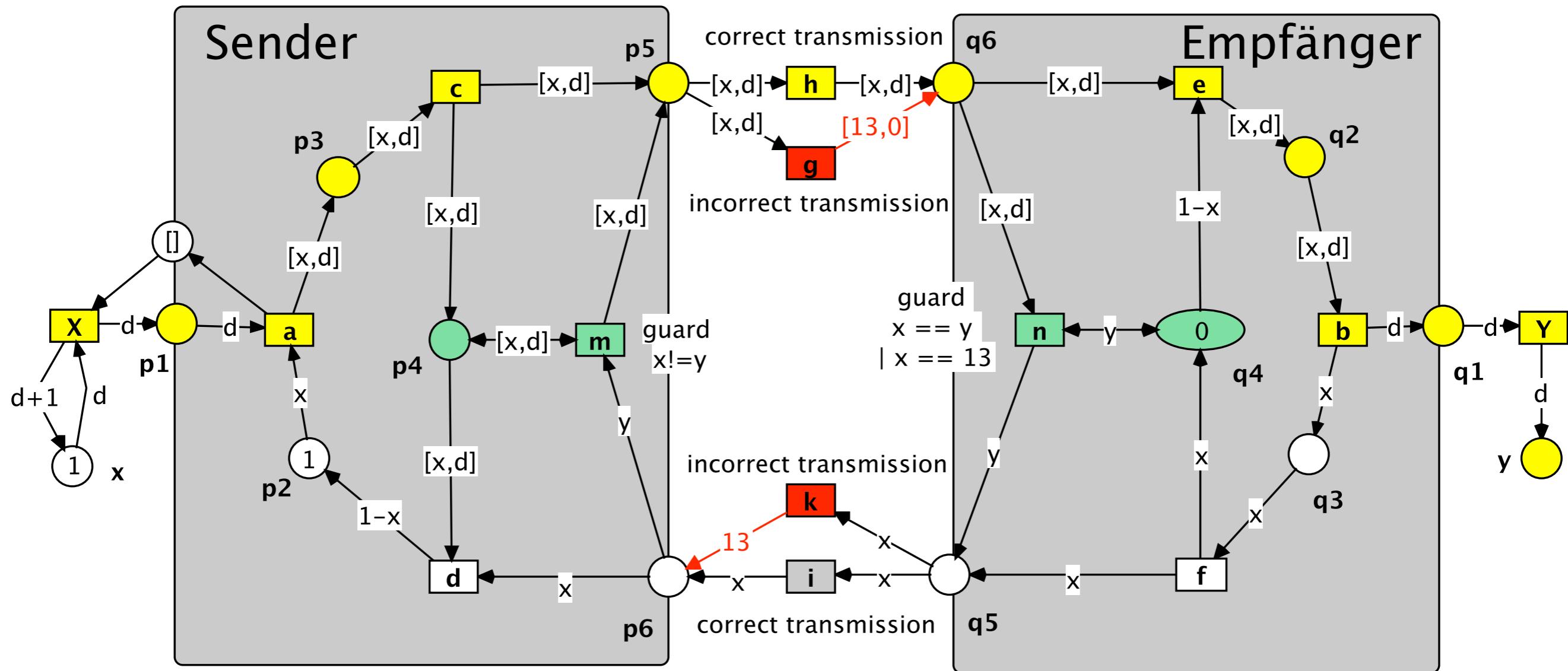
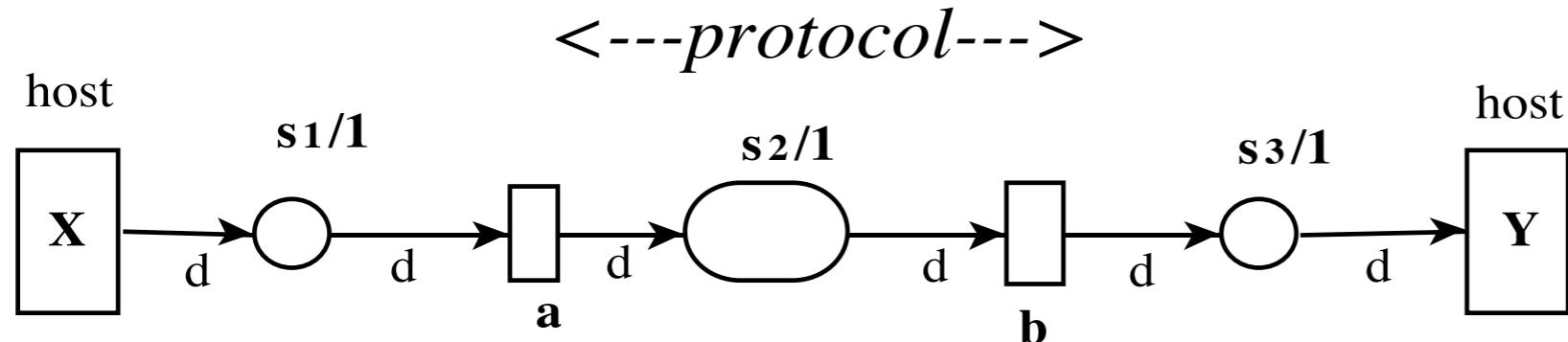


Abbildung 2.60: Das Alternierbitprotokoll abp-5

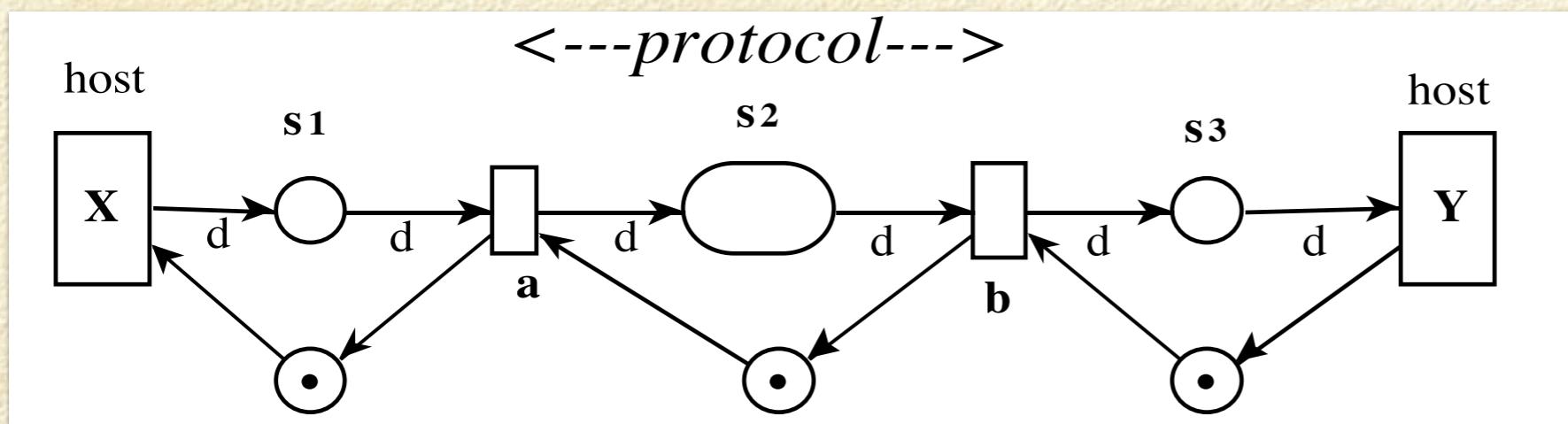


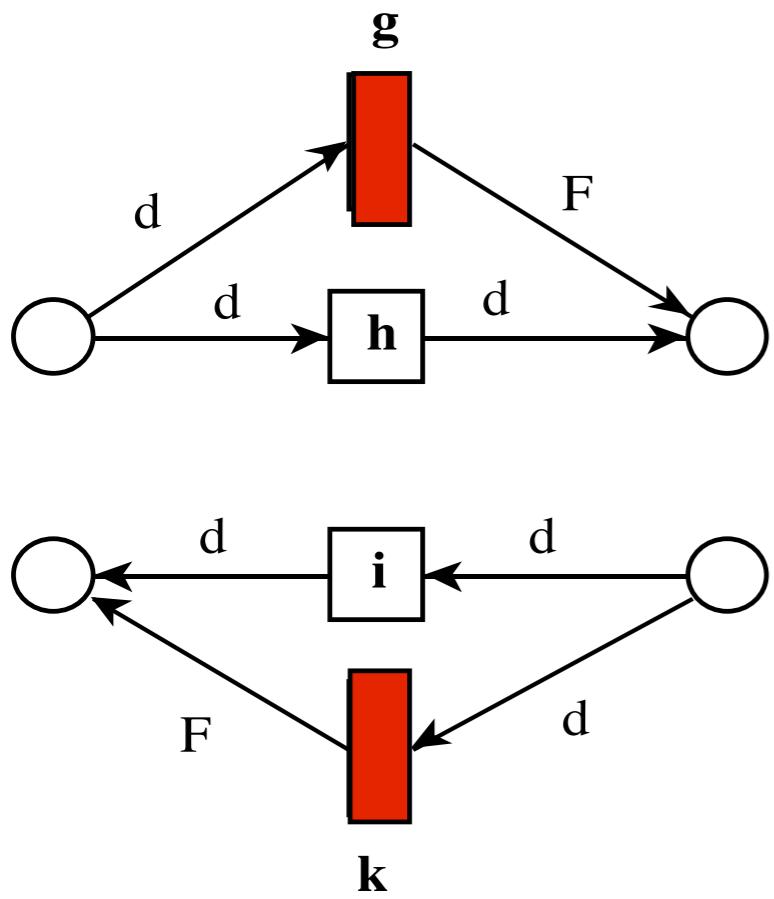
Alternierbit - Protokoll *gefärbtes Netz in RENEW*

Gefärbtes Netz: das Alternierbit-Protokoll

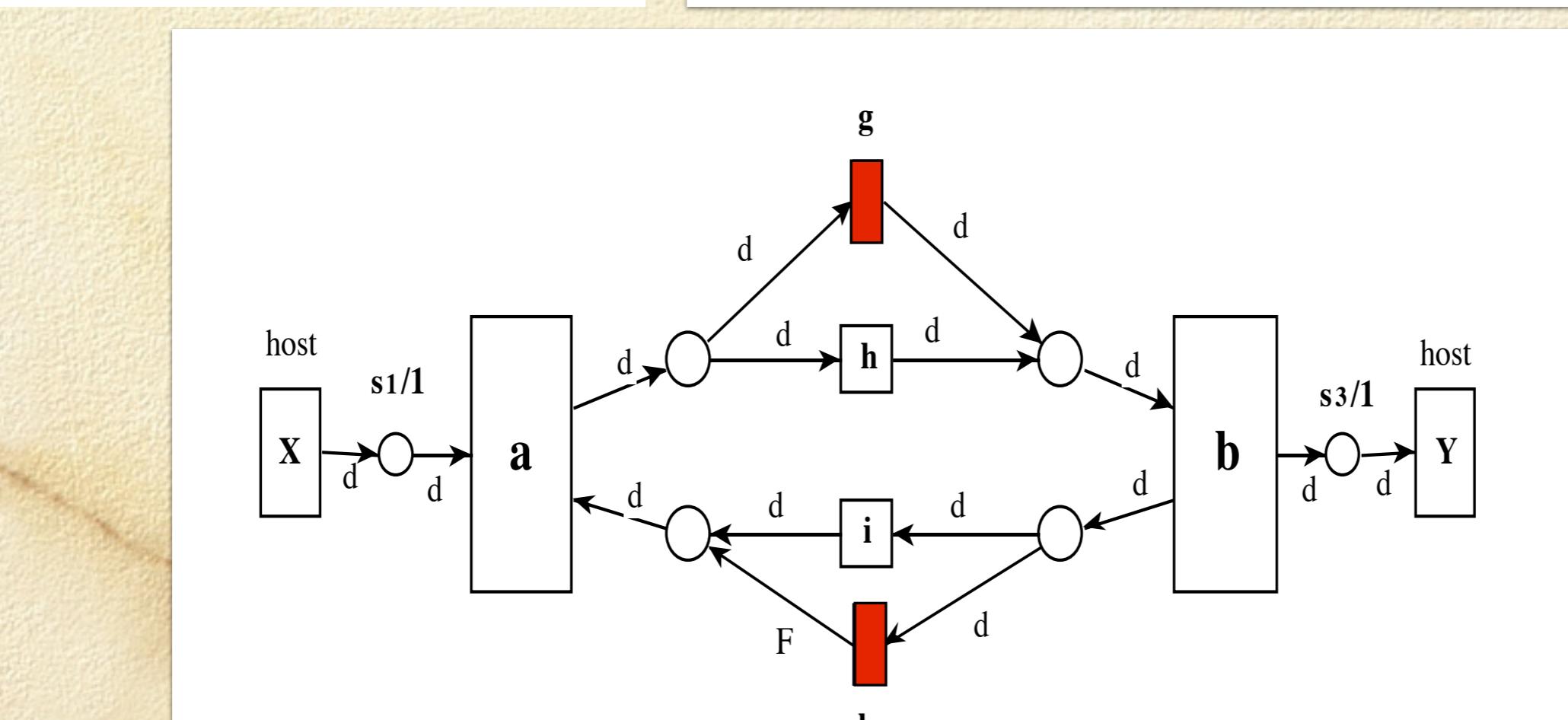


Spezifikation des
Alternierbitprotokolls

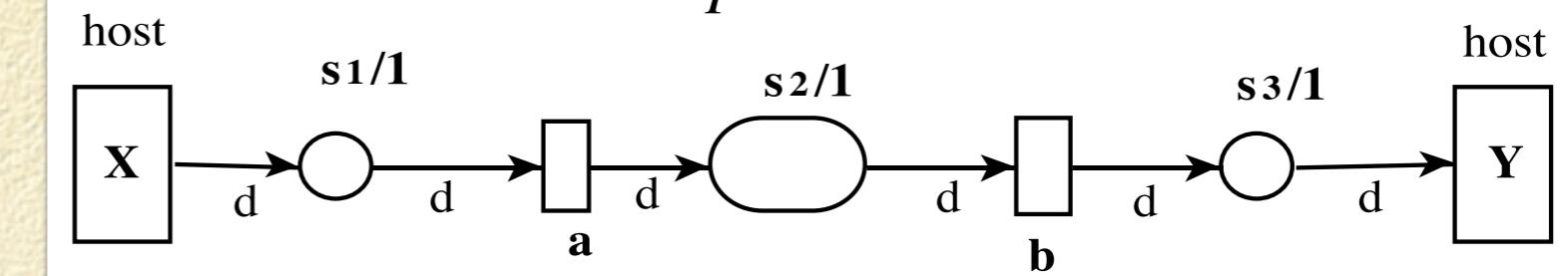


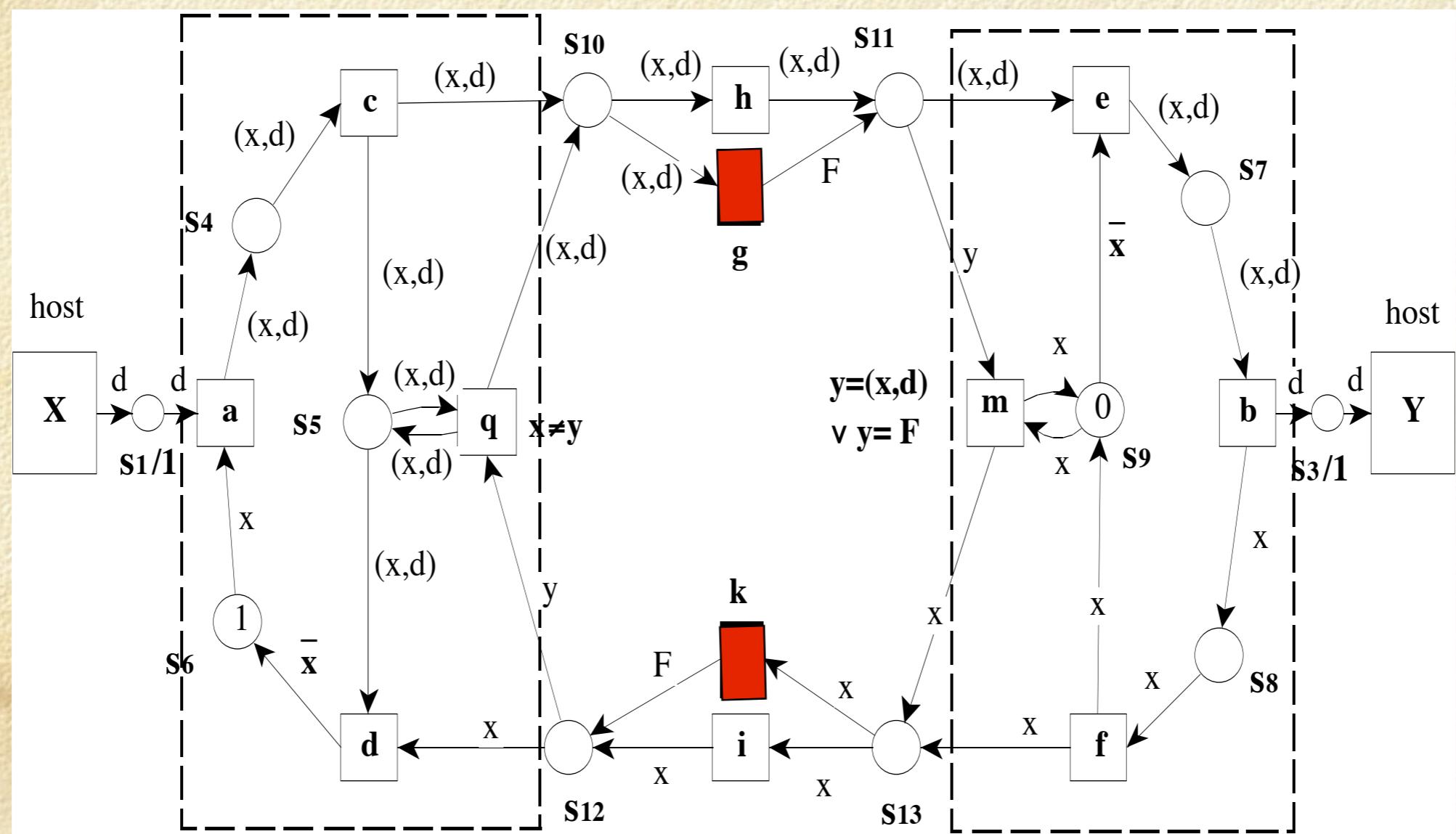
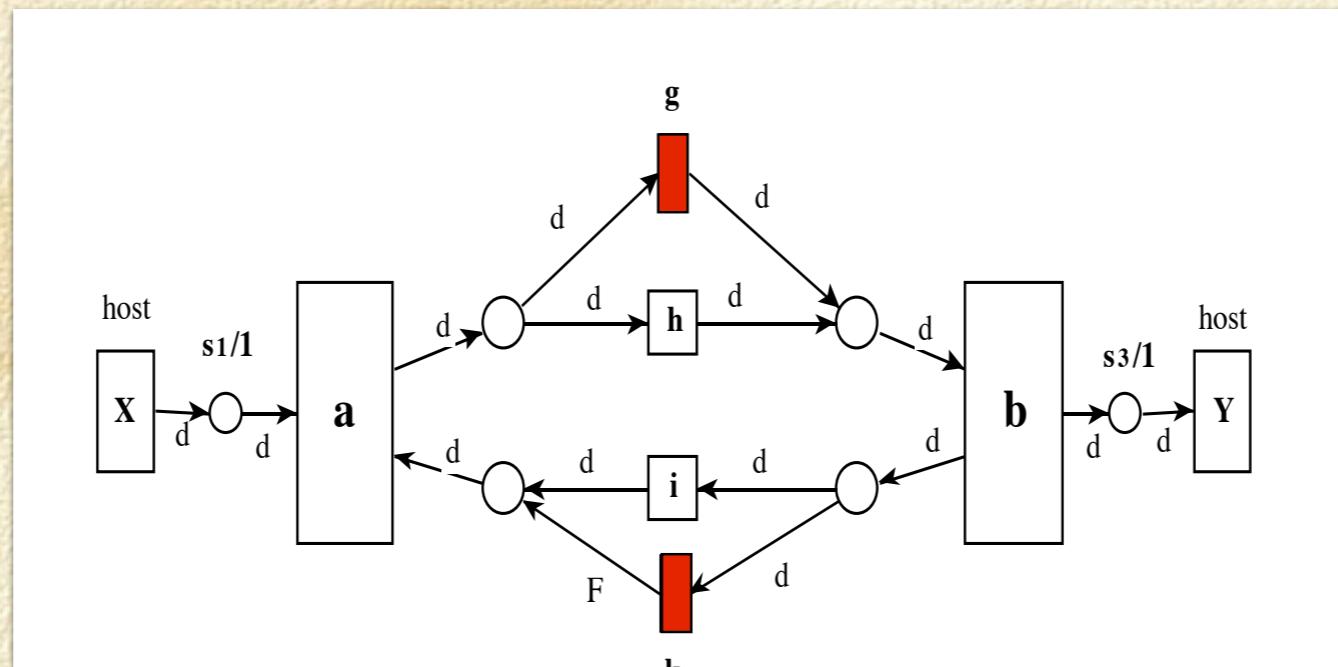


Halbduplexkanal mit Fehlererkennung



Gefärbtes Netz: das Alternierbit-Protokoll

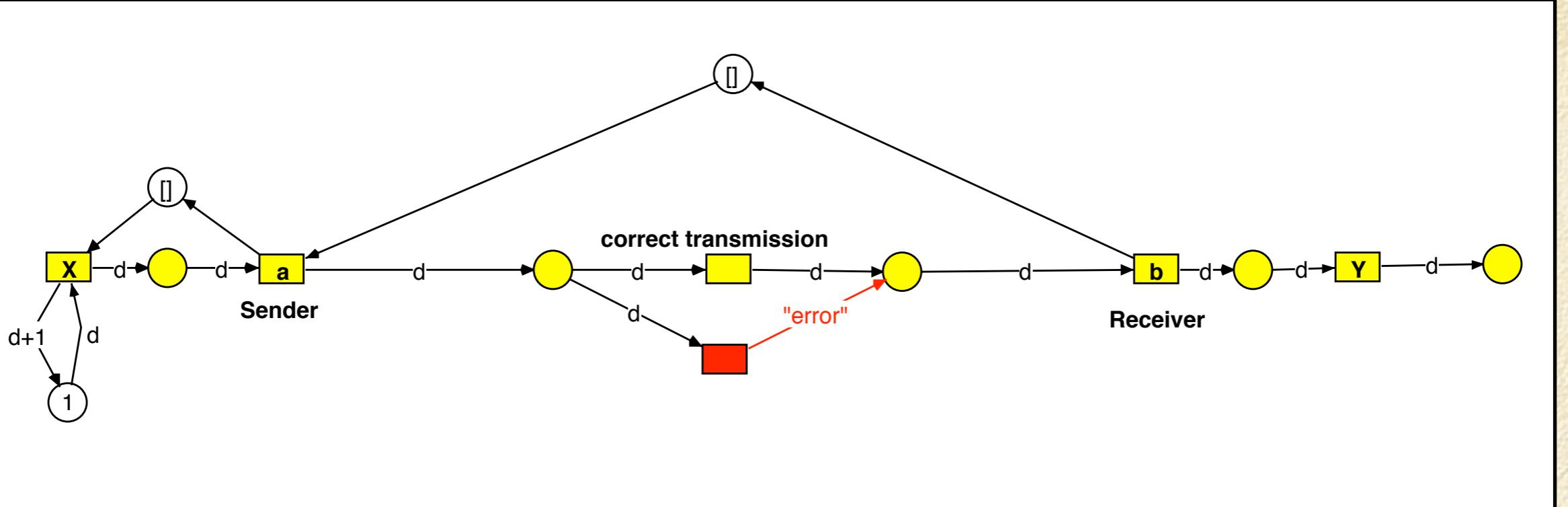
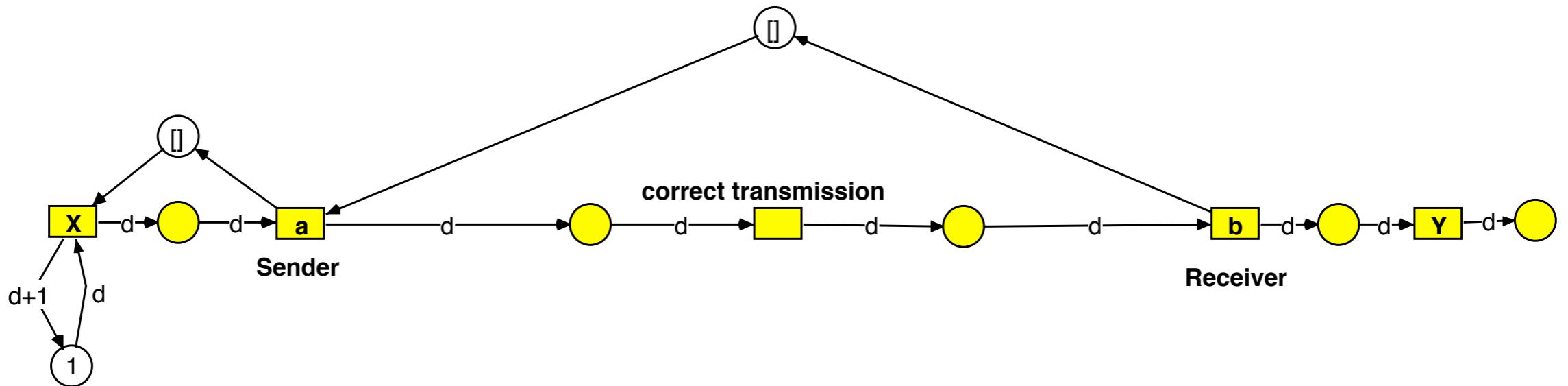




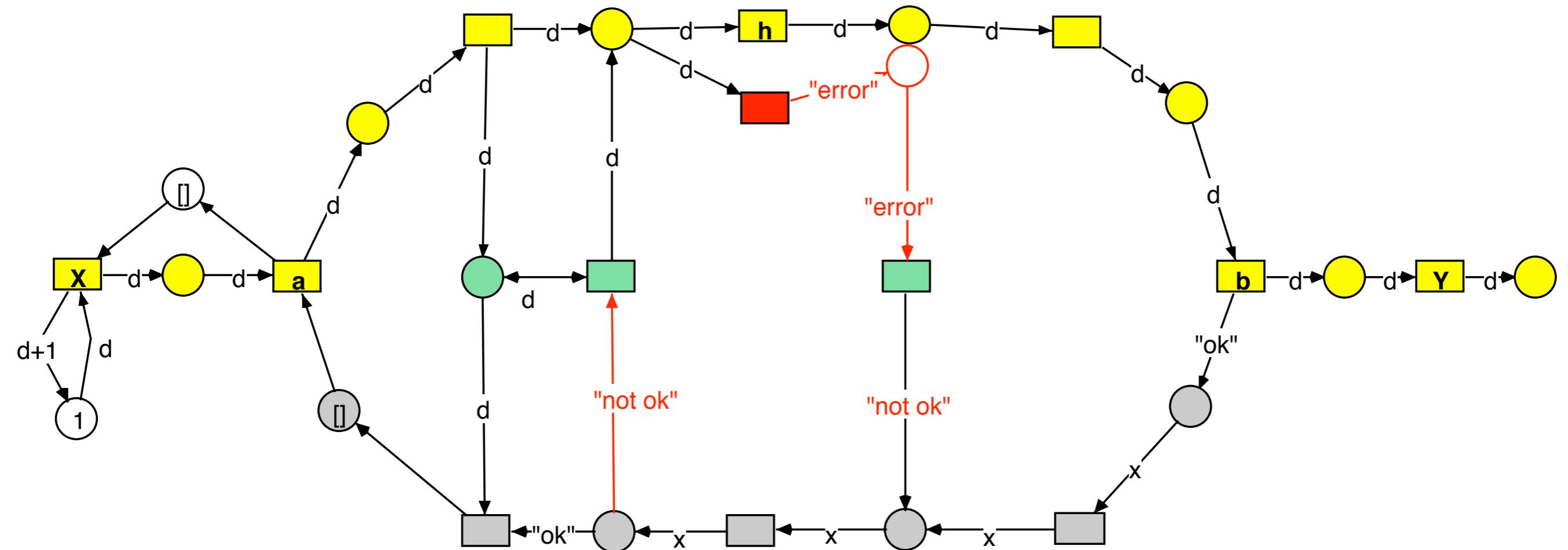


Alternierbit-Protokoll in RENEW

case: the Alternating Bit Protocol



case: the Alternating Bit Protocol



case: the Alternating Bit Protocol

