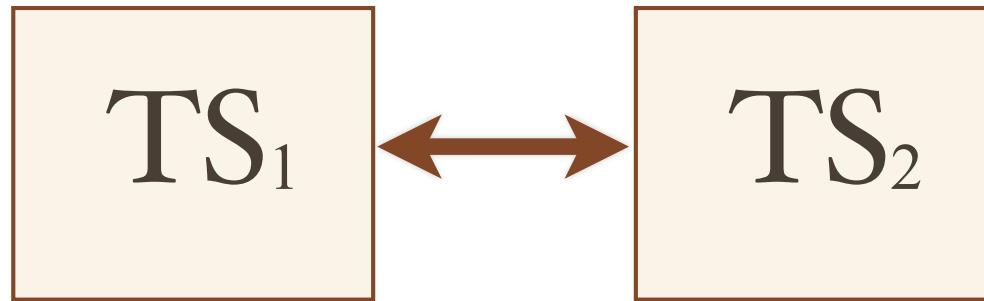


FGI 2

Daniel Moldt

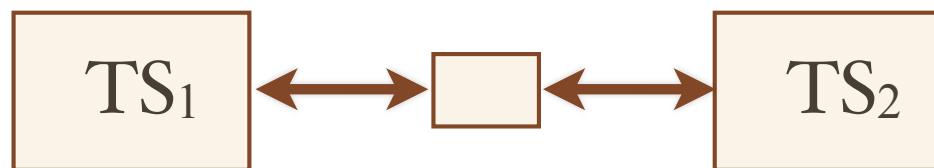
Synchrone Produkte (Wdh.)

Produkte von Transitionssystemen

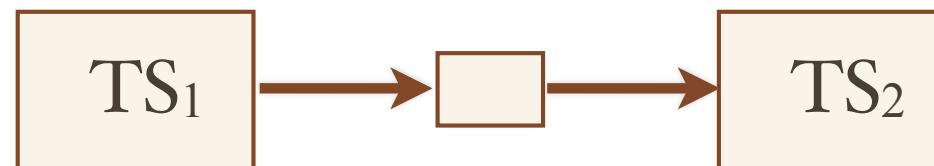


- a) *Rendezvous-Synchronisation*
handshake synchronisation

- b) *Speicher-Synchronisation*

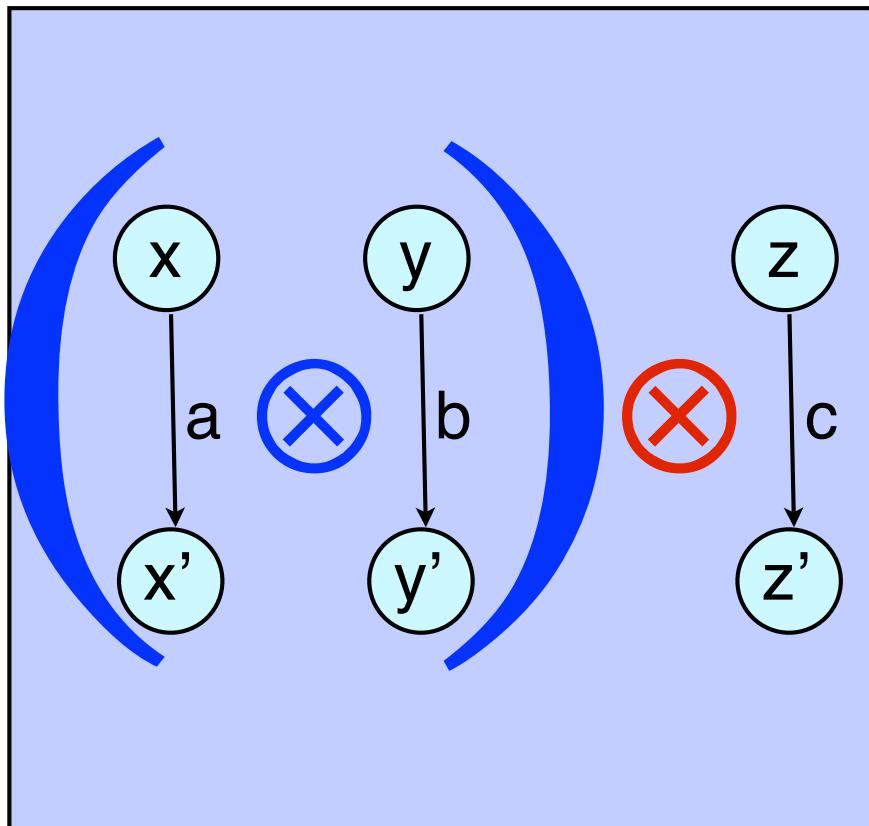


- c) *Nachrichten-Synchronisation*

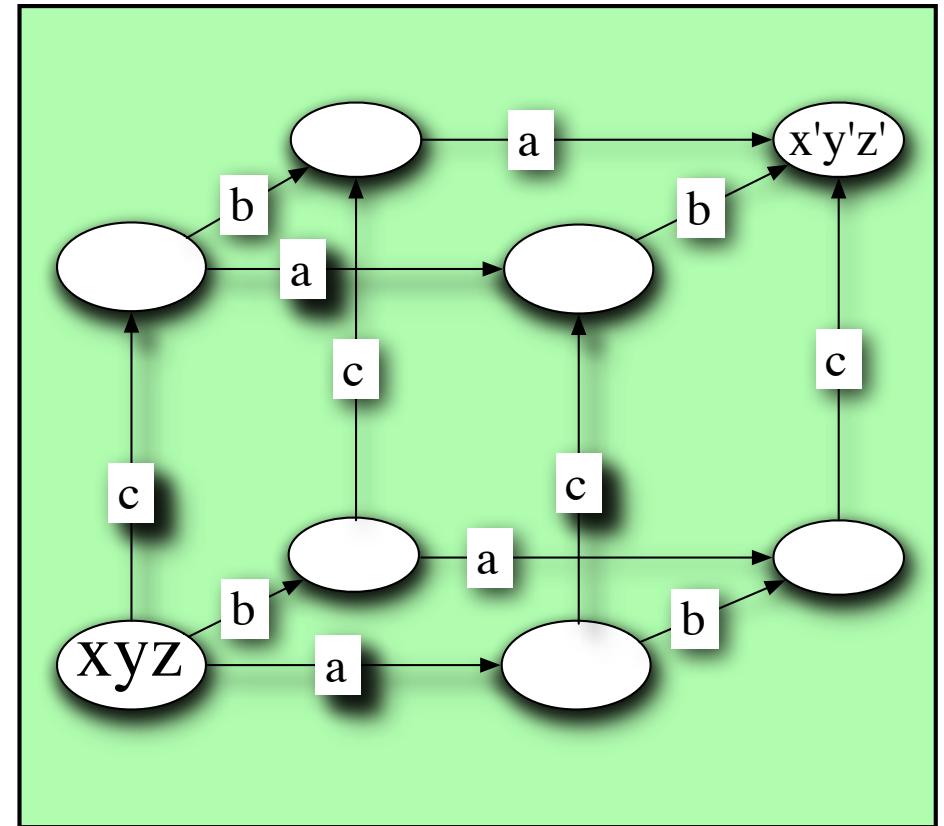


Interleaving von Systemen

Produkt: $(TS_1 \otimes TS_2) \otimes TS_3$



=



Interleaving: $Sync_1 = \emptyset$

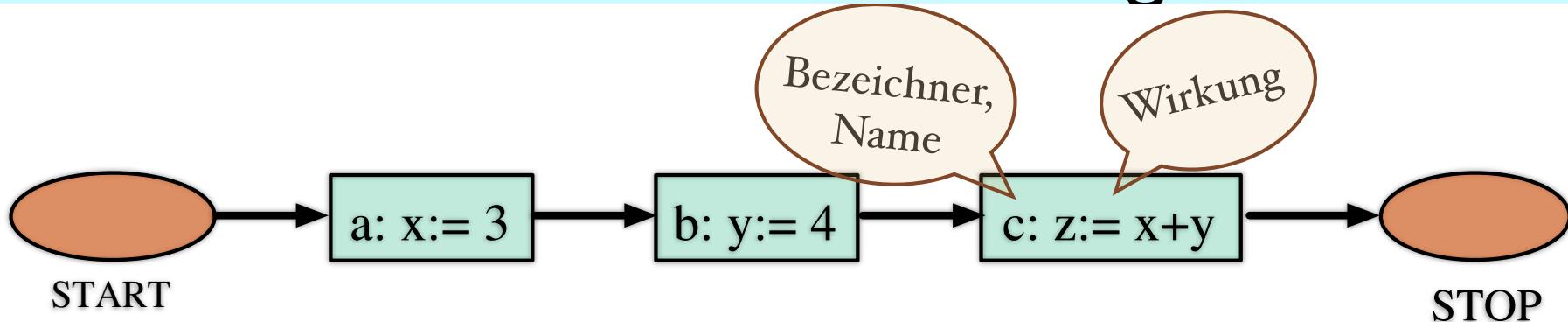
Interleaving: $Sync_2 = \emptyset$

FGI 2

Daniel Moldt

Partial order semantics

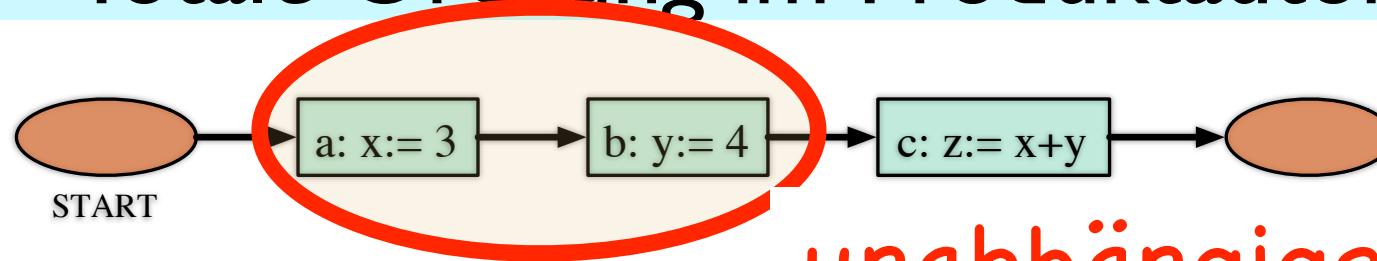
Prozesse als Ordnungen



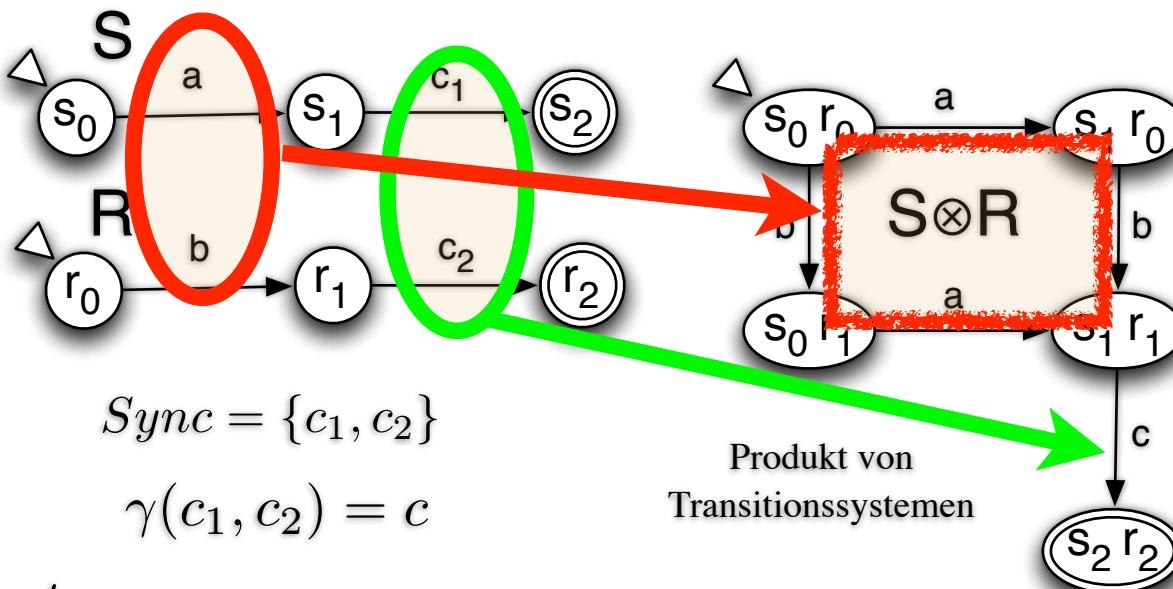
Eigenschaften der Handlungen:

- *extensional*, d.h. durch ihre Wirkung beschreibbar
- *unteilbar* (auch atomar), d.h. sie werden vom Prozessor ununterbrochen ausgeführt
„atomare Operation“
- *geordnet*
 - a) durch eine totale Ordnung: sequentieller Prozess

Totale Ordnung im Produktautomaten



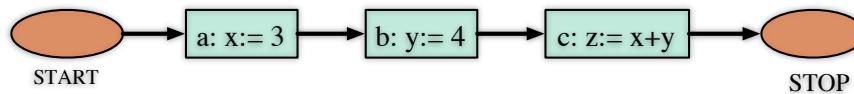
unabhängige Aktionen



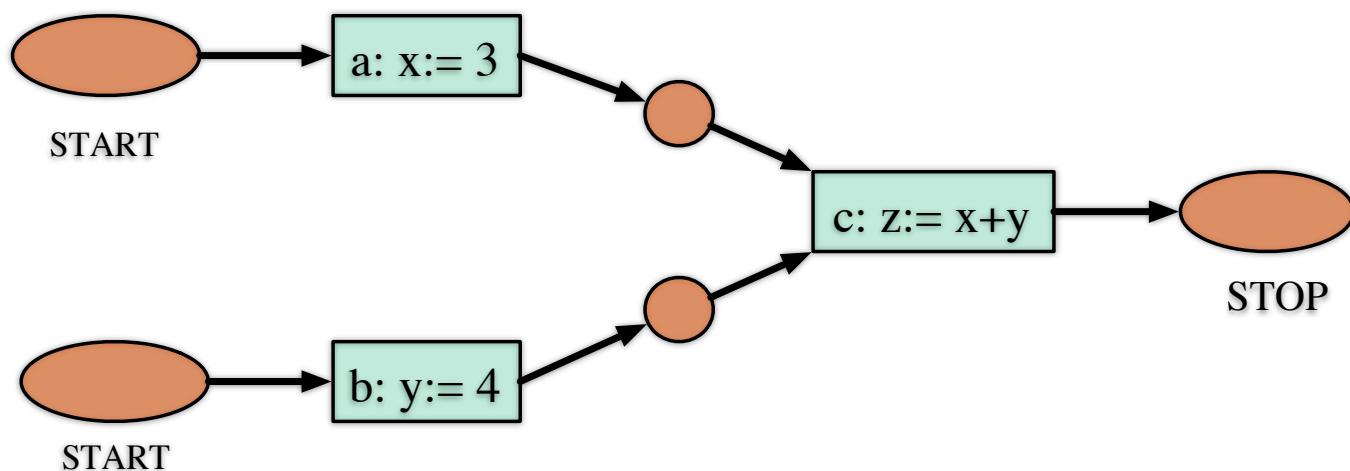
- geordnet

- durch eine totale Ordnung: sequentieller Prozess
- partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

Partielle Ordnung im Petrinetz

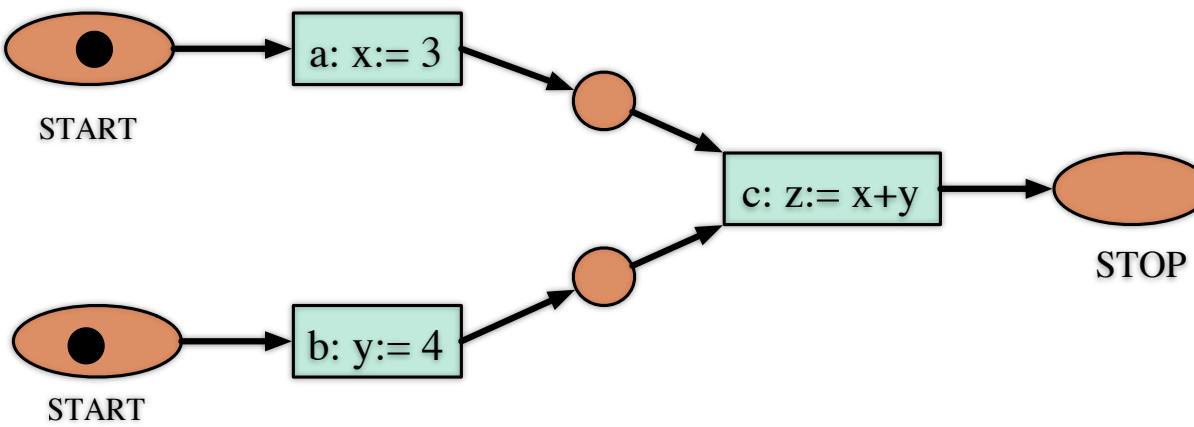


... als Petrinetz:



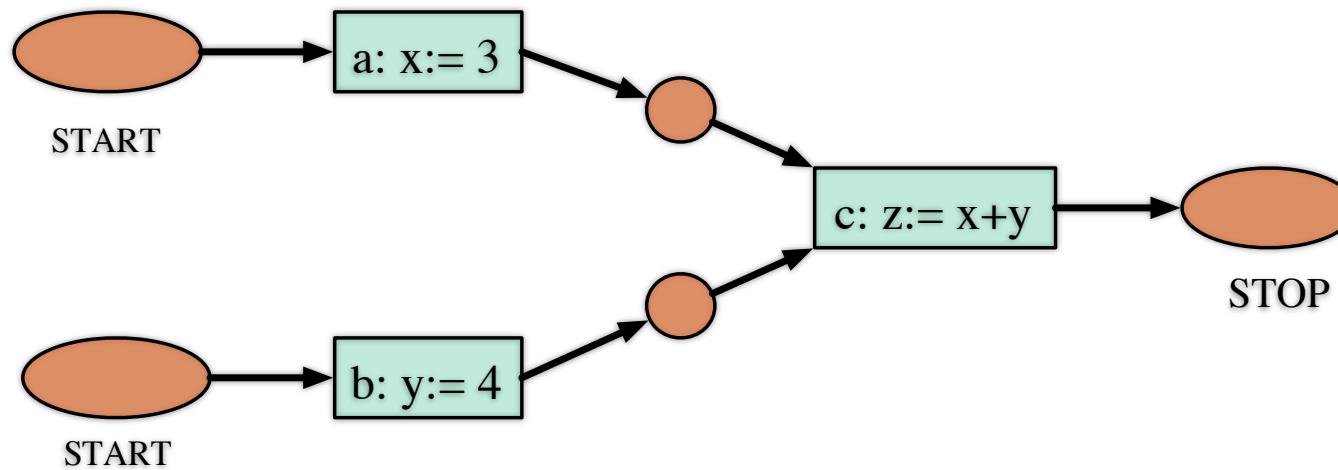
- b) partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

Partielle Ordnung im Petrinetz



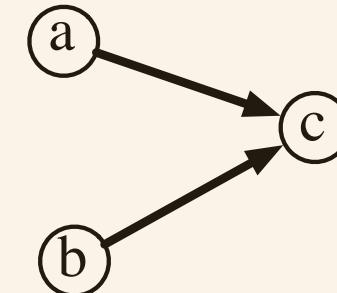
- b) partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

interleaving vs. partial order semantics



als (Aktions-)Folgen:

a;b;c und b;a;c



als Striktordnung
oder partielle Ordnung

interleaving semantics
Folgen-Semantik

partial order semantics
PO-Semantik

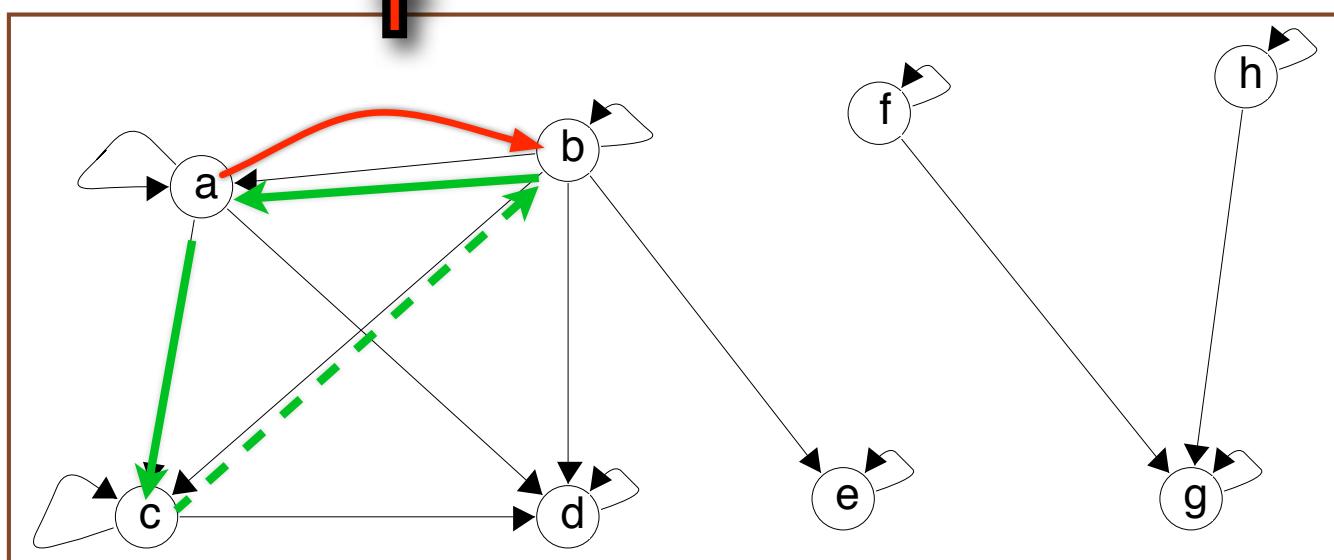
Partielle Ordnungen

Definition 1.21 Sei A eine Menge und $R \subseteq A \times A$ eine (binäre) Relation.

a) (A, R) heißt partielle Ordnung (partially ordered set, poset), falls gilt:

- | | |
|---|------------------|
| 1. $\forall a \in A. (a, a) \in R$ | “Reflexivitat” |
| 2. $\forall a, b \in A. (a, b) \in R \wedge (b, a) \in R \Rightarrow a = b$ | “Antisymmetrie” |
| 3. $\forall a, b, c \in A. (a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R$ | “Transitivitat” |

Schreibweise: $a \leq b$ für $(a, b) \in R$



Zyklen?

keine Zyklen !!!

$$R = \{(a,a), (a,c), (b,a), \dots\}$$

Striktordnungen

Definition 1.21 Sei A eine Menge und $R \subseteq A \times A$ eine (binäre) Relation.

Striktordnung

a) (A, R) heißt partielle Ordnung (partially ordered set, poset), falls gilt:

1. $\forall a \in A. (a, a) \notin R$

Irreflexivität

„Reflexivität“

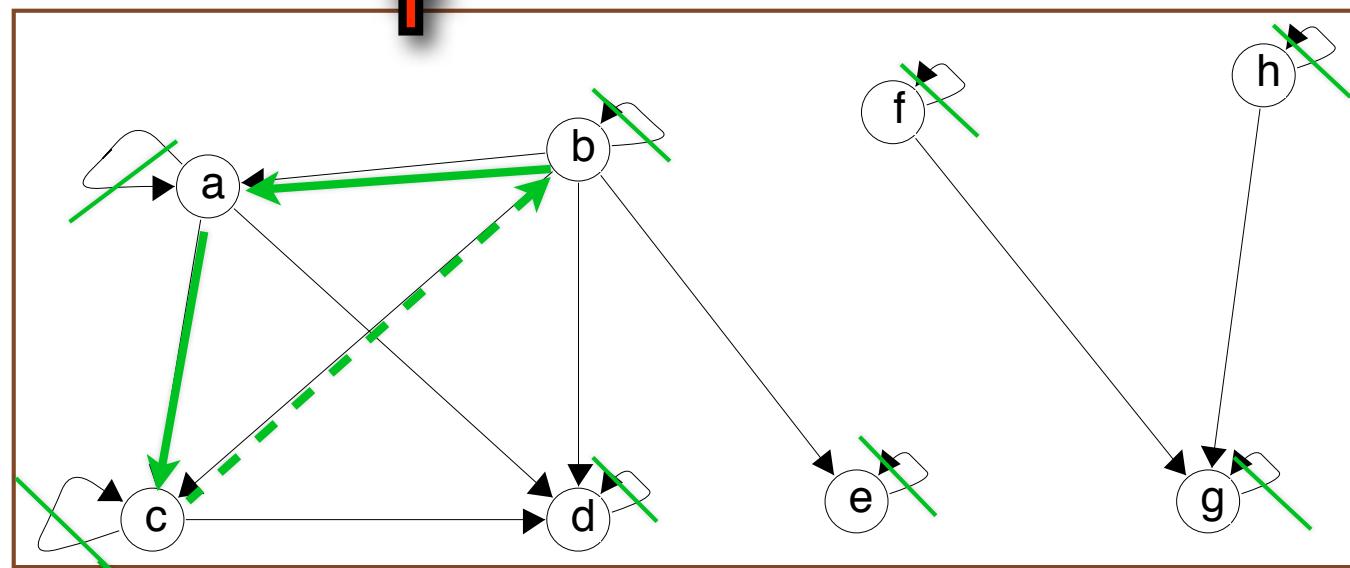
2. $\forall a, b \in A. (a, b) \in R \wedge (b, a) \in R \Rightarrow a = b$

„Antisymmetrie“

3. $\forall a, b, c \in A. (a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R$

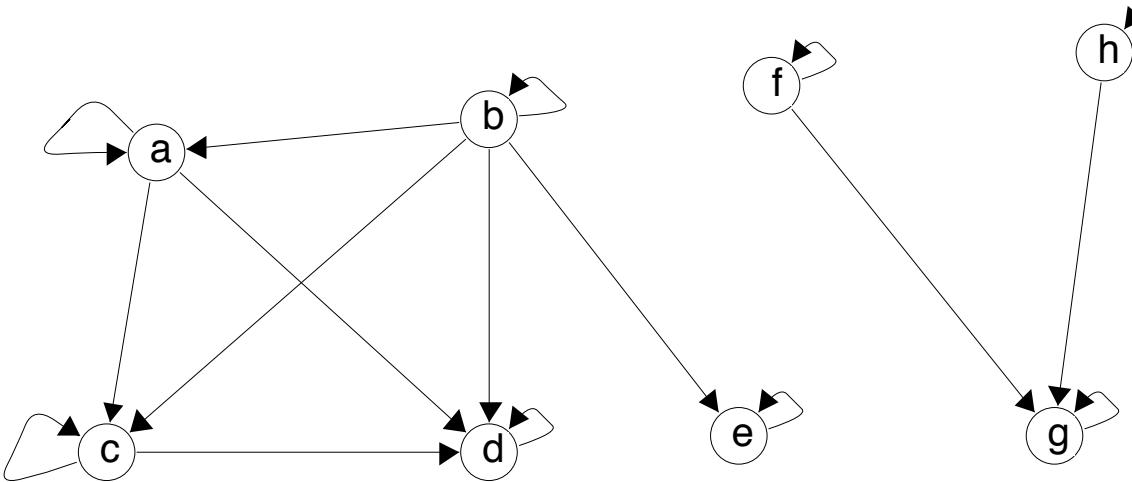
„Transitivität“

Schreibweise: $a < b$ für $(a, b) \in R$



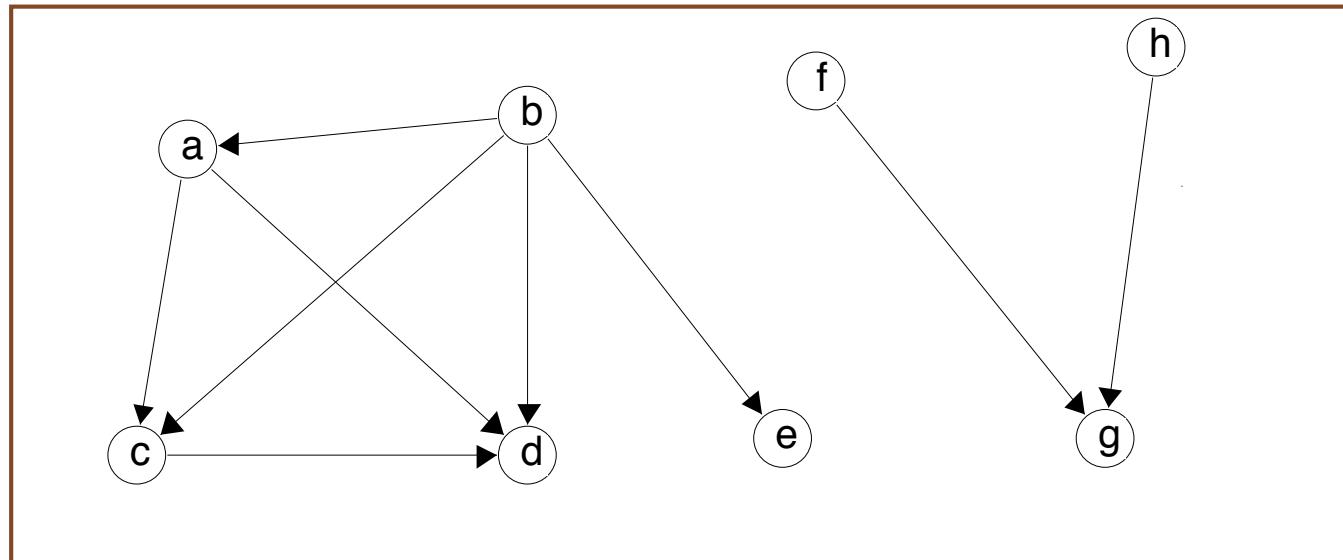
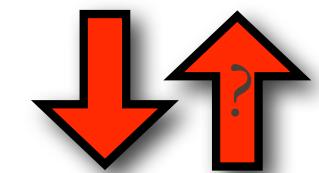
... redundant
wg. 1.+3.

$$R = \{(a,a), (a,c), (b,a), \dots\}$$



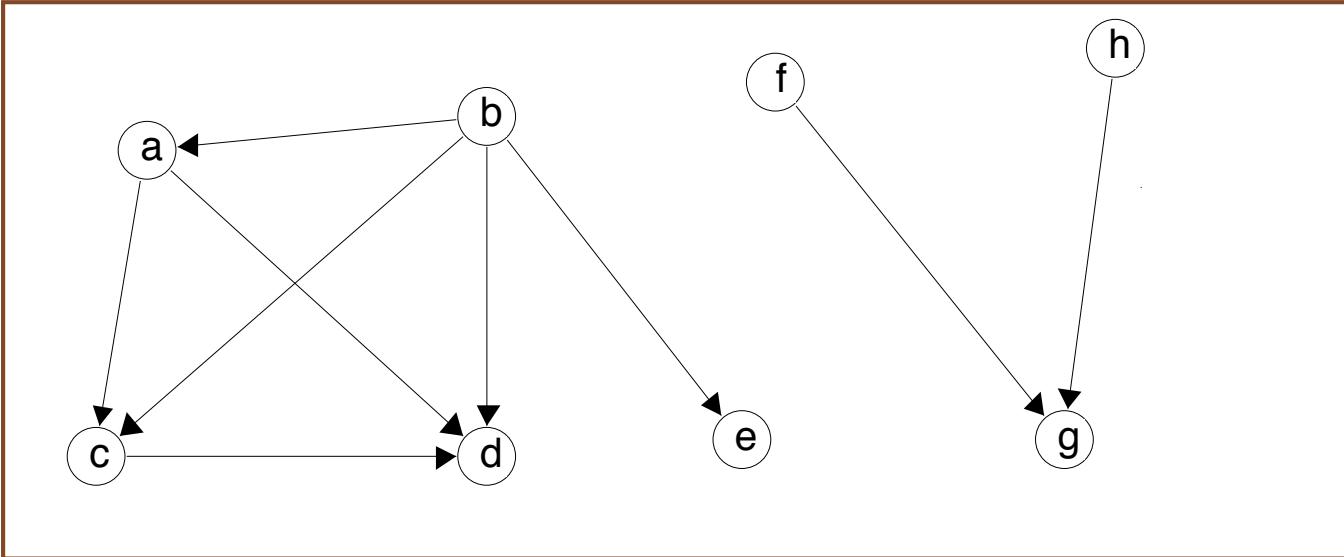
*partielle
Ordnung*

$$R = \{(a,a), (a,c), (b,a), \dots\}$$



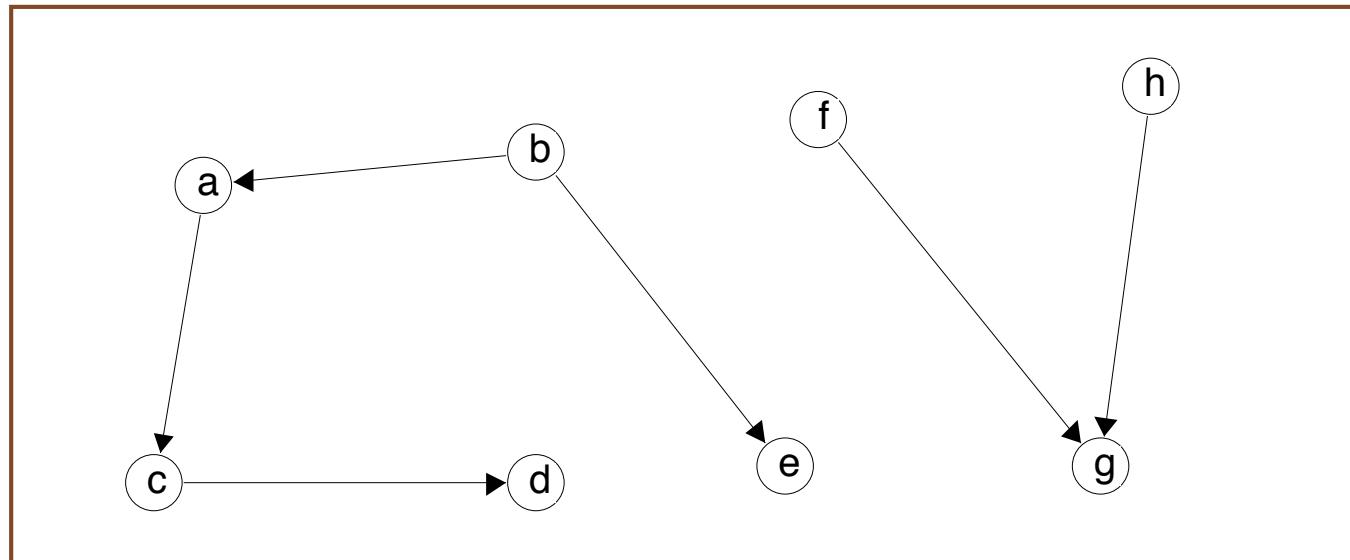
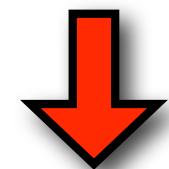
*strikte
Ordnung*

$$S = R - id = \{(b,a), (a,c), (b,c), \dots\}$$

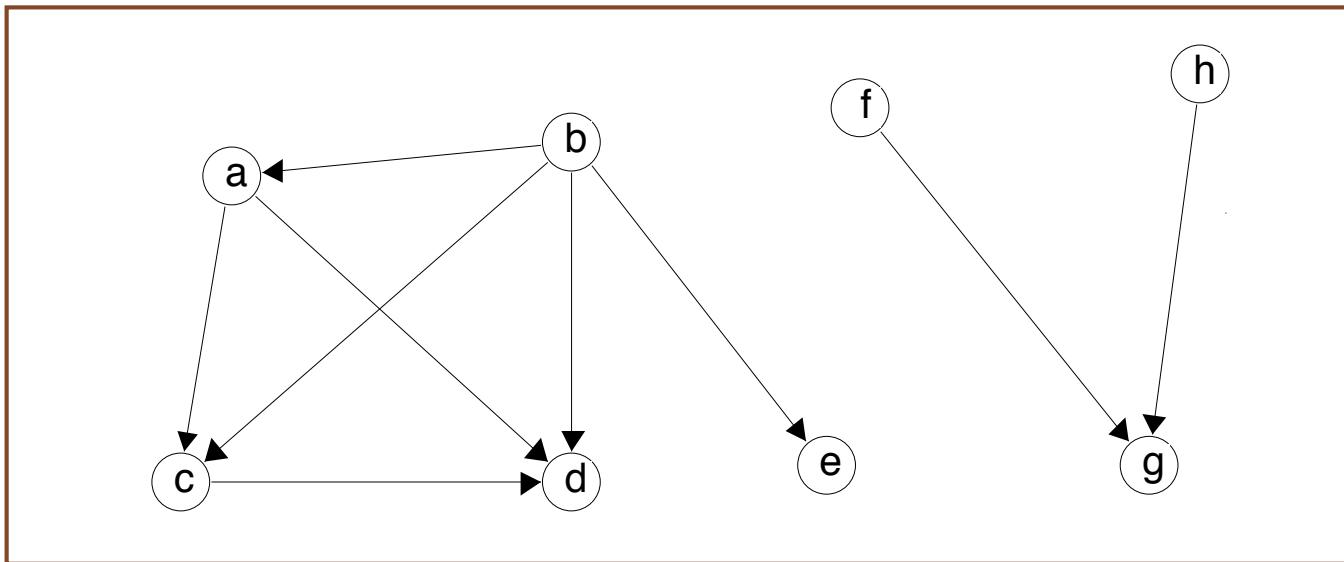


*strikte
Ordnung*

$$S = R - id = \{(b,a), (a,c), (b,c), \dots\}$$



*Präzedenz-
Relation*



*strikte
Ordnung*

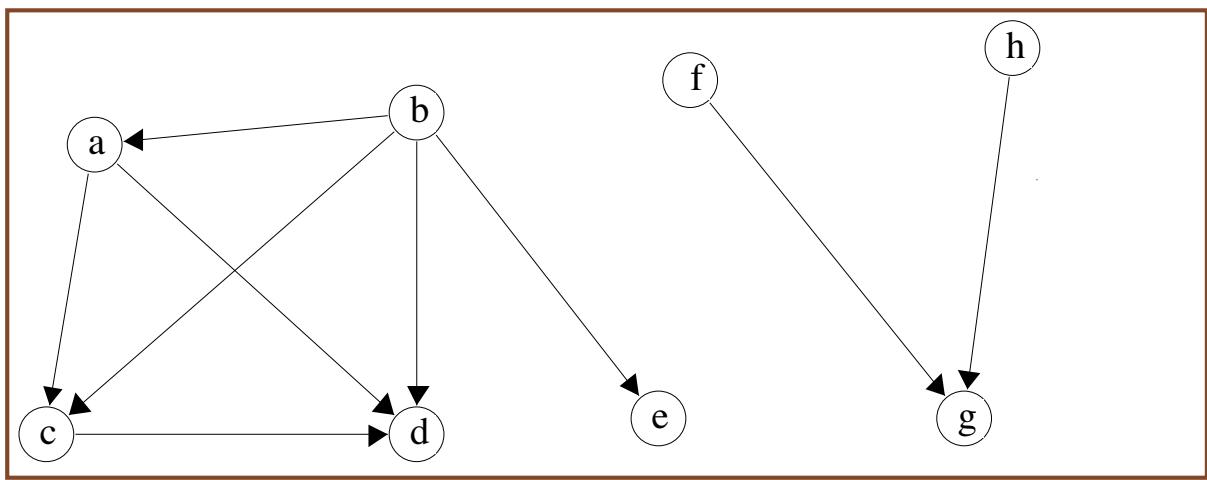
$$S = R - id = \{(b,a), (a,c), (b,c), \dots\}$$

1. b heißt **direkter Nachfolger** von a (in Zeichen: $a \lessdot b$), falls:

$$a \lessdot b : \Leftrightarrow a < b \wedge \neg \exists c \in A. a < c \wedge c < b$$

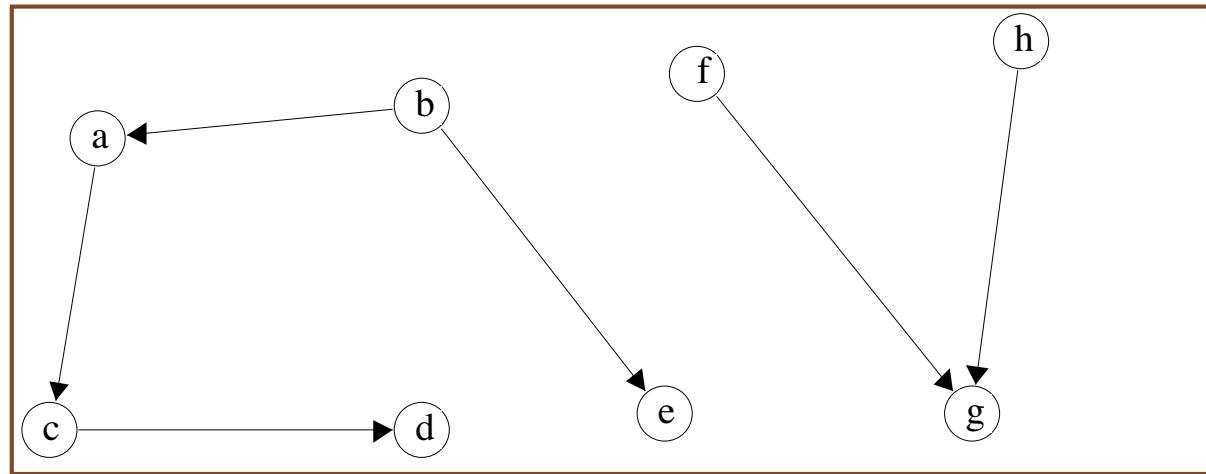
*keiner
dazwischen*

2. (A, \lessdot) heißt **Präzedenzrelation** zu $(A, <)$.



*strikte
Ordnung*

$$S = R - id_A = \{(b.a), (a,c), (c,d), (\cancel{a,d}), (\cancel{b,c}), (\cancel{b,d}), (b,e), (f,g), (h,g)\}$$



*Präzedenz-
Ordnung*

$$Q = S - S^2 = \{(b.a), (a,c), (c,d), (\cancel{a,d}), (\cancel{b,c}), (\cancel{b,d}), (b,e), (f,g), (h,g)\}$$

$$<^2 := < \circ <= \{(a,b) \mid \exists c. a < c \wedge c < b\}$$

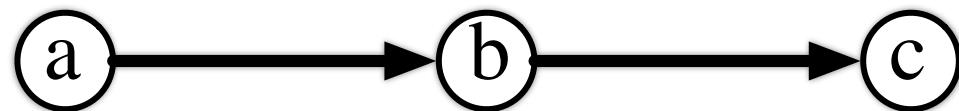
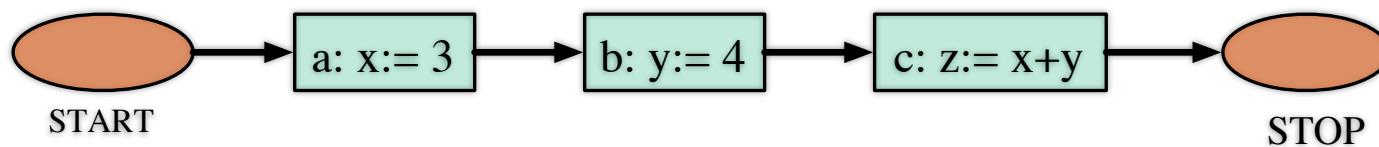
Lineare Ordnungen

c) (A, R) heißt totale oder lineare Ordnung (totally ordered set), falls gilt:

1. (A, R) ist partielle Ordnung

2. $\forall a, b \in A. a \neq b \text{ impliziert } (a, b) \in R \vee (b, a) \in R$

“Vollständigkeit”

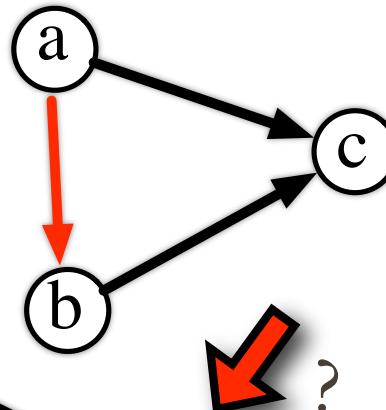


Linearisierung einer Striktordnung

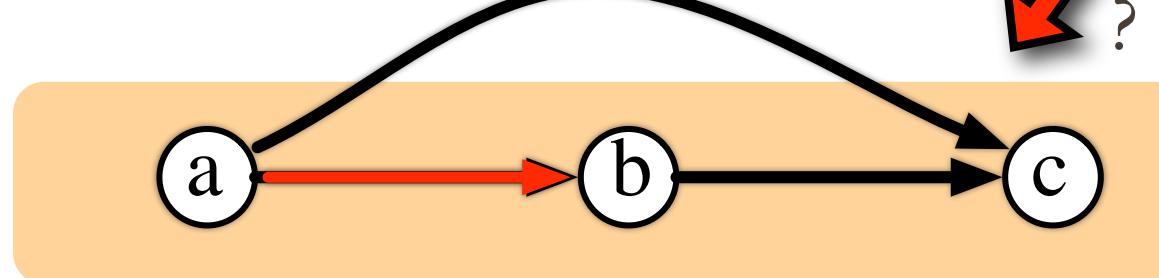
a;b;c und b;a;c

interleaving semantics

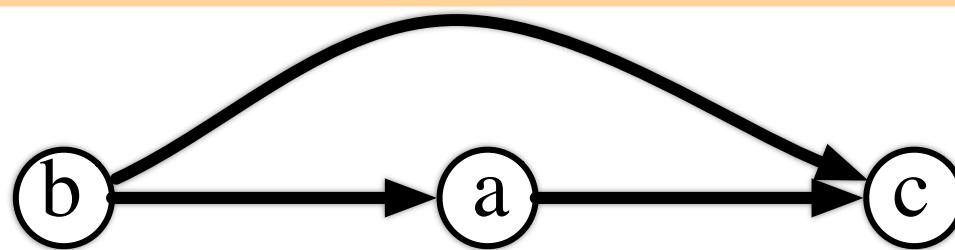
Folgen-Semantik



$\{(a,c), (b,c)\}$



$\{(a,b), (a,c), (b,c)\}$



$\{(b,a), (a,c), (b,c)\}$

e) Für eine Striktordnung $(A, <)$ ist

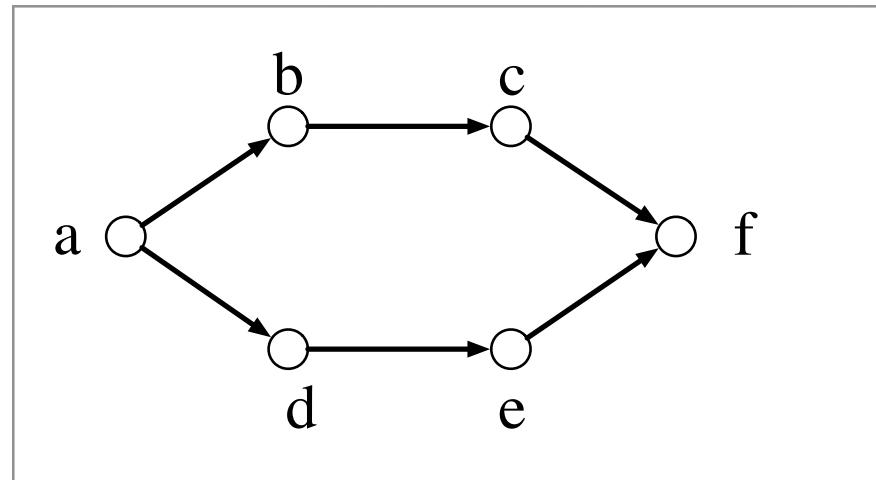
$Lin(A, <) := \{(A, <_1) | <_1$ ist lineare Striktordnung mit $< \subseteq <_1\}$

die Menge der linearen (oder seriellen) Vervollständigungen von

$(A, <).$

Beispiel #1

Beispiel: (A, \lessdot)

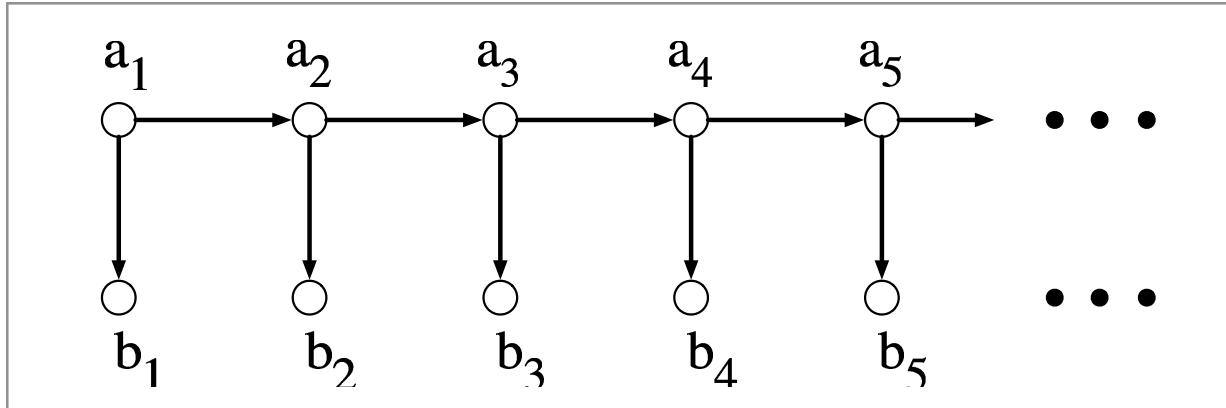


$$\begin{aligned} Lin(A, \lessdot) = \{ & \quad a \quad b \quad c \quad d \quad e \quad f \quad , \\ & \quad a \quad b \quad d \quad c \quad e \quad f \quad , \\ & \quad a \quad d \quad b \quad c \quad e \quad f \quad , \\ & \quad a \quad d \quad b \quad e \quad c \quad f \quad , \\ & \quad a \quad d \quad e \quad b \quad c \quad f \quad , \\ & \quad a \quad b \quad d \quad e \quad c \quad f \quad \} \end{aligned}$$

Konstruktionsprinzip?

Beispiel #2

Beispiel: (A, \lessdot)



$$\begin{aligned} Lin(A, \lessdot) = \{ & \quad a_1 \quad b_1 \quad a_2 \quad b_2 \quad a_3 \quad b_3 \quad a_4 \quad b_4 \quad a_5 \quad b_5 \quad \dots \\ & a_1 \quad a_2 \quad b_1 \quad b_2 \quad a_3 \quad b_3 \quad a_4 \quad b_4 \quad a_5 \quad b_5 \quad \dots \\ & a_1 \quad a_2 \quad b_1 \quad a_3 \quad b_2 \quad b_3 \quad a_4 \quad b_4 \quad a_5 \quad b_5 \quad \dots \\ & \vdots \end{aligned}$$

Konstruktionsprinzip?



FGI 2

Daniel Moldt

Zeitstempel in verteilten Systemen

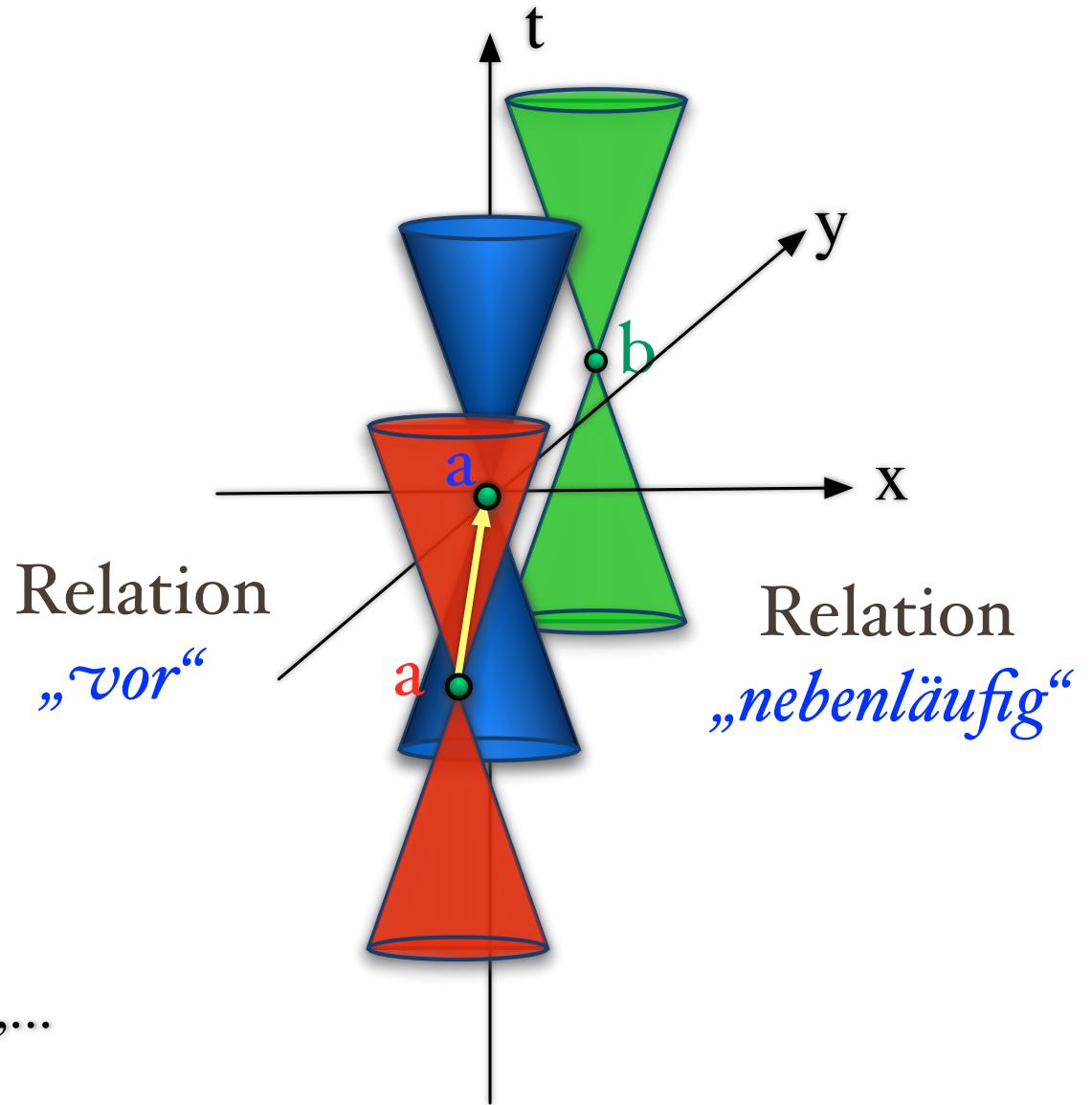
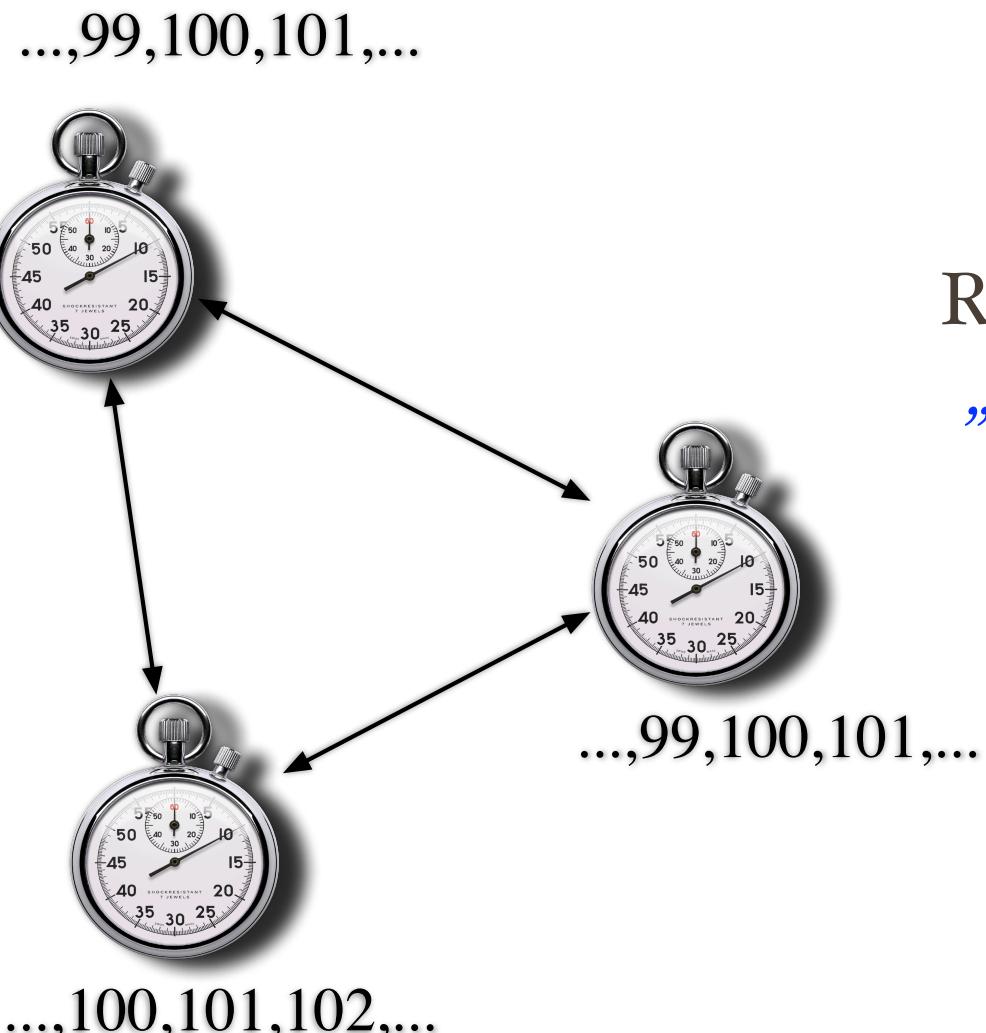
Logische und vektorielle Zeitstempel

globale Zeit

vs.

lokale Zeit

Minkowski-Diagramm wurde [1908](#) von [Hermann Minkowski](#) entwickelt



Definition 5.6 Ein Nachrichten-Modell ist ein System von n Funktionseinheiten bzw. Prozessoren p_0, \dots, p_{n-1} , die

- lokale Rechenschritte ausführen und
- Nachrichten an andere versenden.

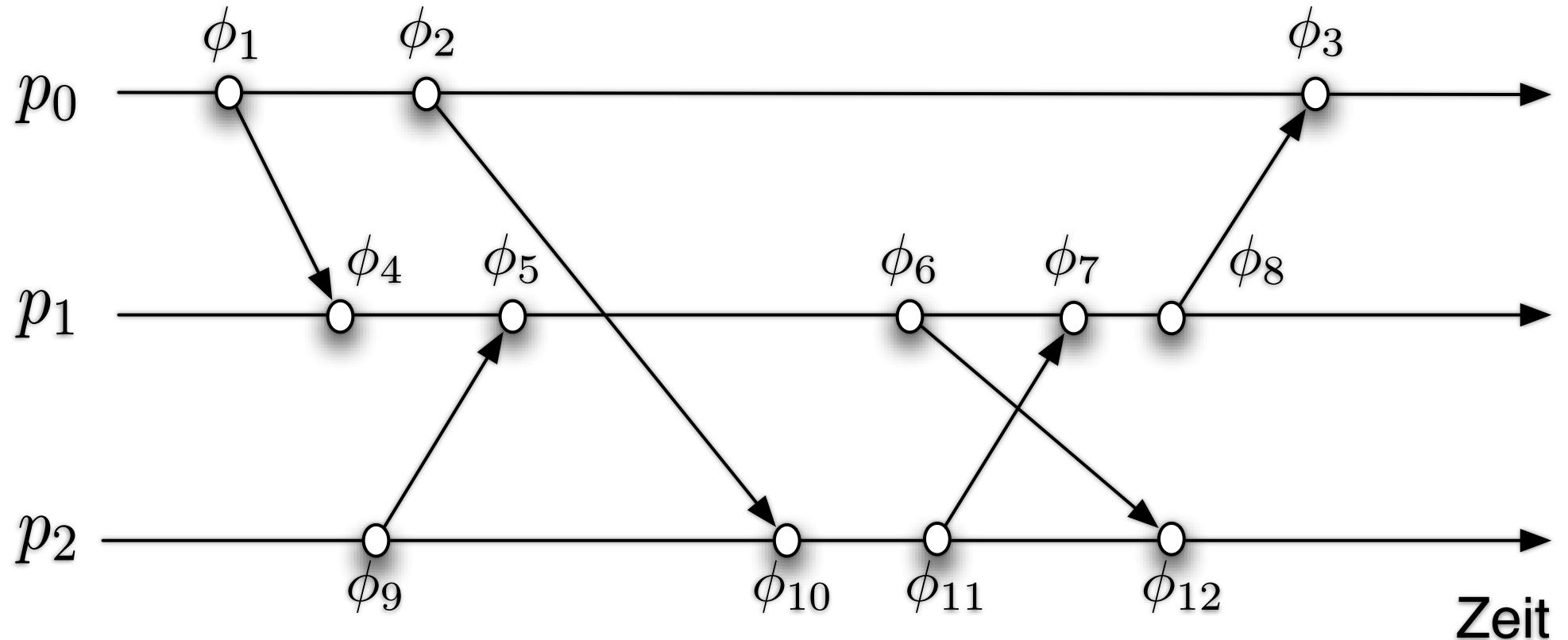
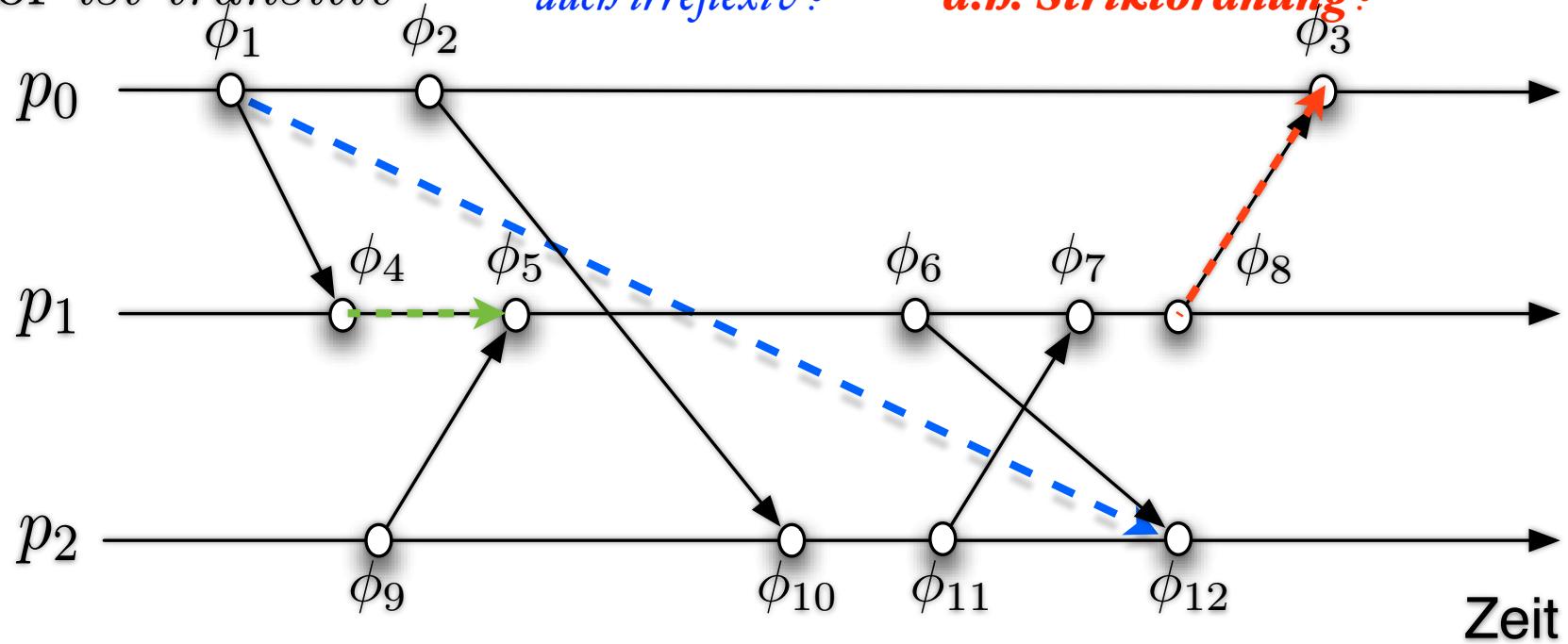


Abbildung 5.4: Nachrichten-Modell mit $n=3$ Prozessoren durch lokale Zeitskalen

Definition 5.7 Es wird eine Relation $\text{vor} \subseteq \Phi \times \Phi$ definiert. Für $\phi_1, \phi_2 \in \Phi$ gelte $(\phi_1 \text{ vor } \phi_2)$, falls folgendes gilt:

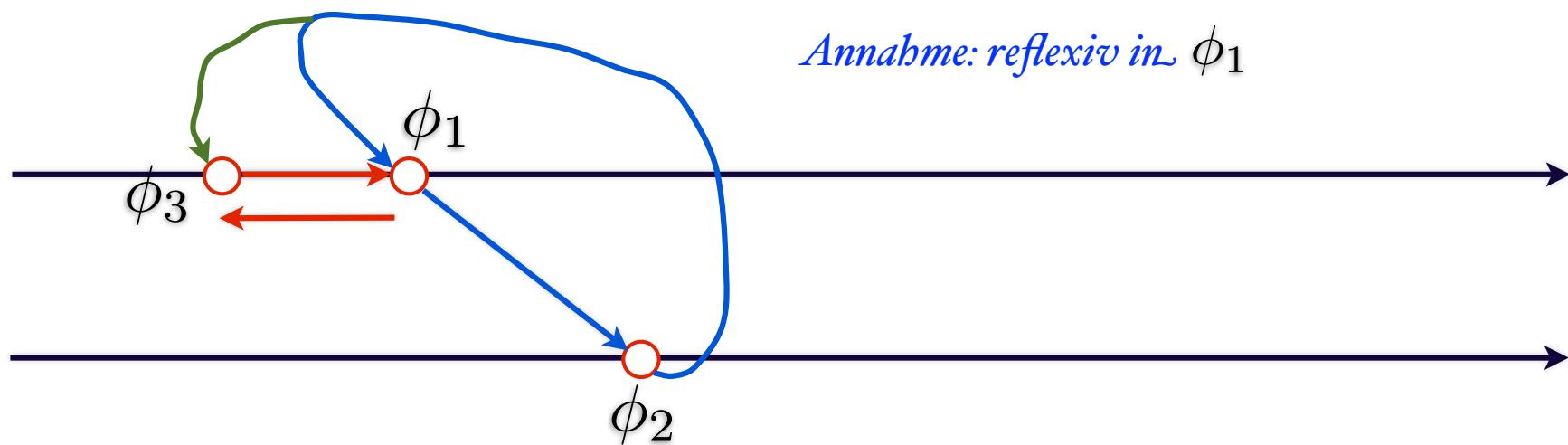
- a) Gehören ϕ_1 und ϕ_2 zu dem **selben** Prozessor (d.h. sie liegen auf der selben (linear geordneten) Zeitachse), dann gilt $(\phi_1 \text{ vor } \phi_2)$ genau dann, wenn ϕ_1 vor ϕ_2 auf der Zeitachse liegt.
- b) Gehören ϕ_1 und ϕ_2 zu **verschiedenen** Prozessoren (d.h. sie liegen auf verschiedenen Zeitachsen) und ist ϕ_1 das Sendeereignis einer Nachricht, die in ϕ_2 empfangen wird, dann gilt $(\phi_1 \text{ vor } \phi_2)$.

- c) vor ist transitiv *auch irreflexiv?* **d.h. Striktordnung?**



Definition 5.7 Es wird eine Relation $\text{vor} \subseteq \Phi \times \Phi$ definiert. Für $\phi_1, \phi_2 \in \Phi$ gelte $(\phi_1 \text{ vor } \phi_2)$, falls folgendes gilt:

- a) Gehören ϕ_1 und ϕ_2 zu dem **selben** Prozessor (d.h. sie liegen auf der selben Zeitachse), dann gilt $(\phi_1 \text{ vor } \phi_2)$ genau dann, wenn ϕ_1 vor ϕ_2 auf der Zeitachse liegt.
- b) Gehören ϕ_1 und ϕ_2 zu **verschiedenen** Prozessoren (d.h. sie liegen auf verschiedenen Zeitachsen) und ist ϕ_1 das Sendeereignis einer Nachricht, die in ϕ_2 empfangen wird, dann gilt $(\phi_1 \text{ vor } \phi_2)$.
- c) vor ist transitiv *auch irreflexiv?* **d.h. Striktordnung?**



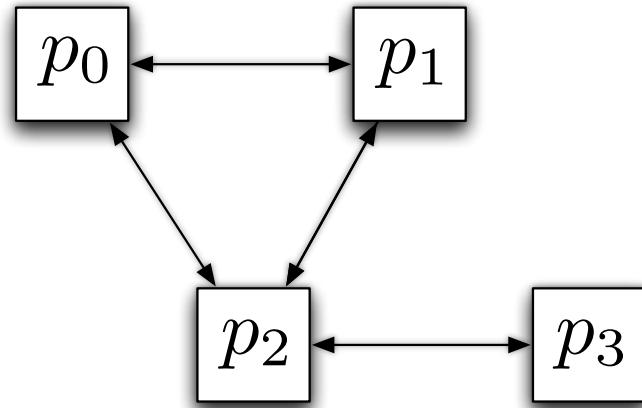
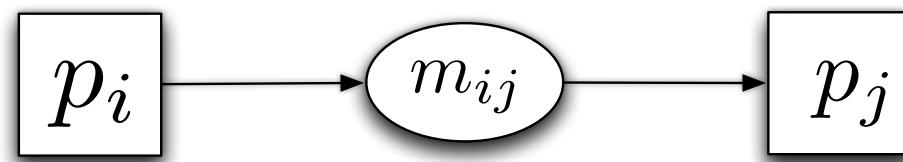
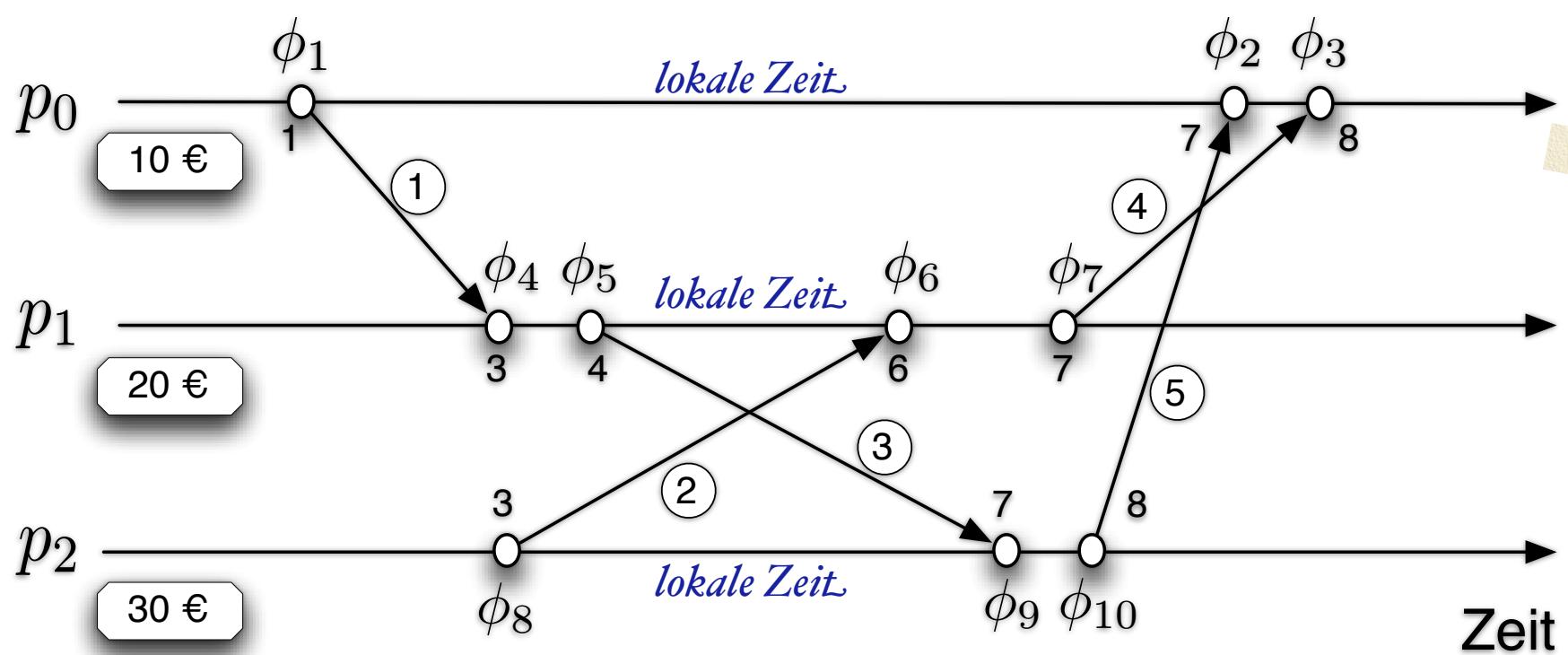


Abbildung 5.5 Banksystem mit 4 Filialen p_0, \dots, p_3



$$x_i := x_i - m_{ij} \qquad \qquad x_j := x_j + m_{ij}$$

Abbildung 5.6 Übertragung von Nachrichten über Kanal

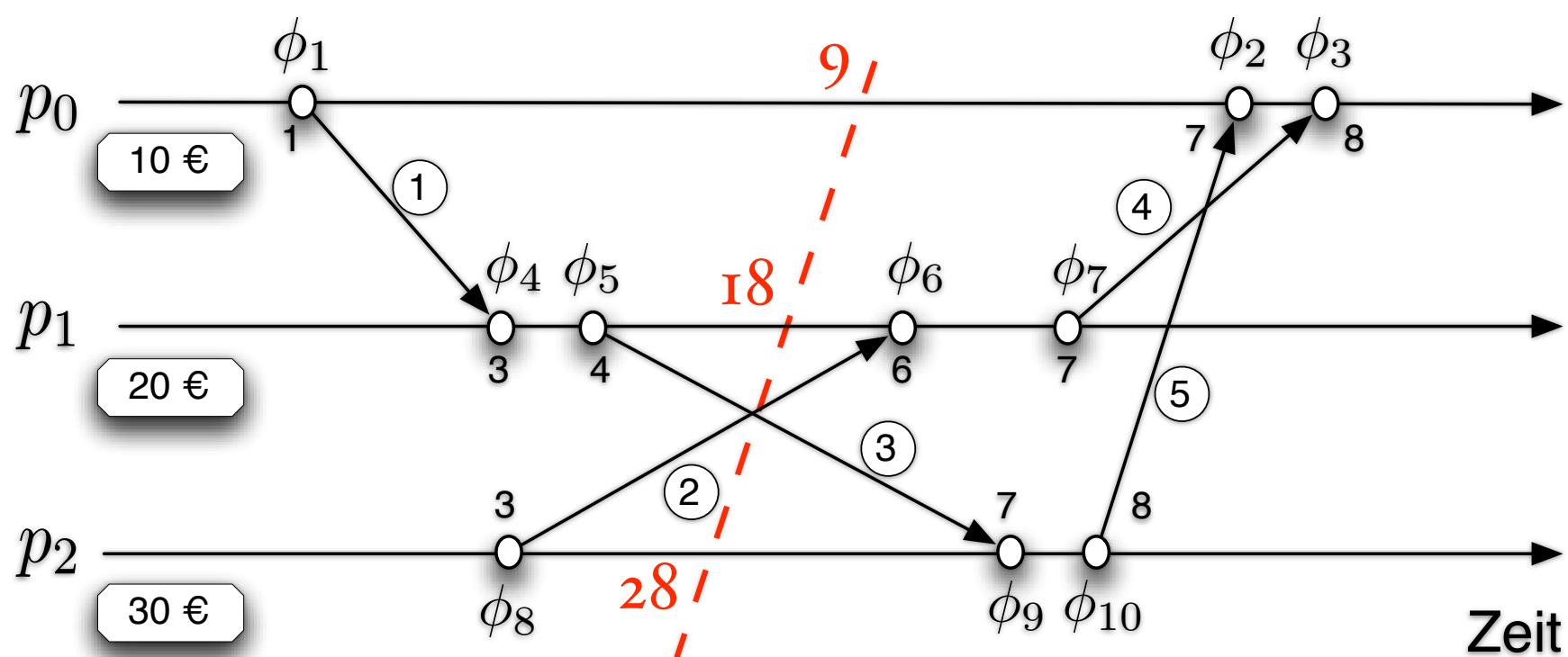


$$\text{Gesamtgeldmenge : } 10 + 20 + 30 = 60$$

$$m(\phi_8, \phi_6) = 2$$

1, 2, 3, ..., 10 sind die Zeitzustände der lokalen Uhren

Problem : Die Bankleitung möchte immer wieder in Intervallen die insgesamt umlaufende Geldmenge ermitteln. Diese sollte immer gleich 60 sein.



Verfahren 1:

Die Bankleitung fordert alle Funktionseinheiten auf, die Kontostände zu einem bestimmten Zeitpunkt t ihrer lokalen Zeit mitzuteilen.

Erwartung : $\sum = 60$.

Defizit: Nachrichten unterwegs

p_0 : 1 an p_1

p_1 : 1 von p_0 ; 3 an p_2 ;

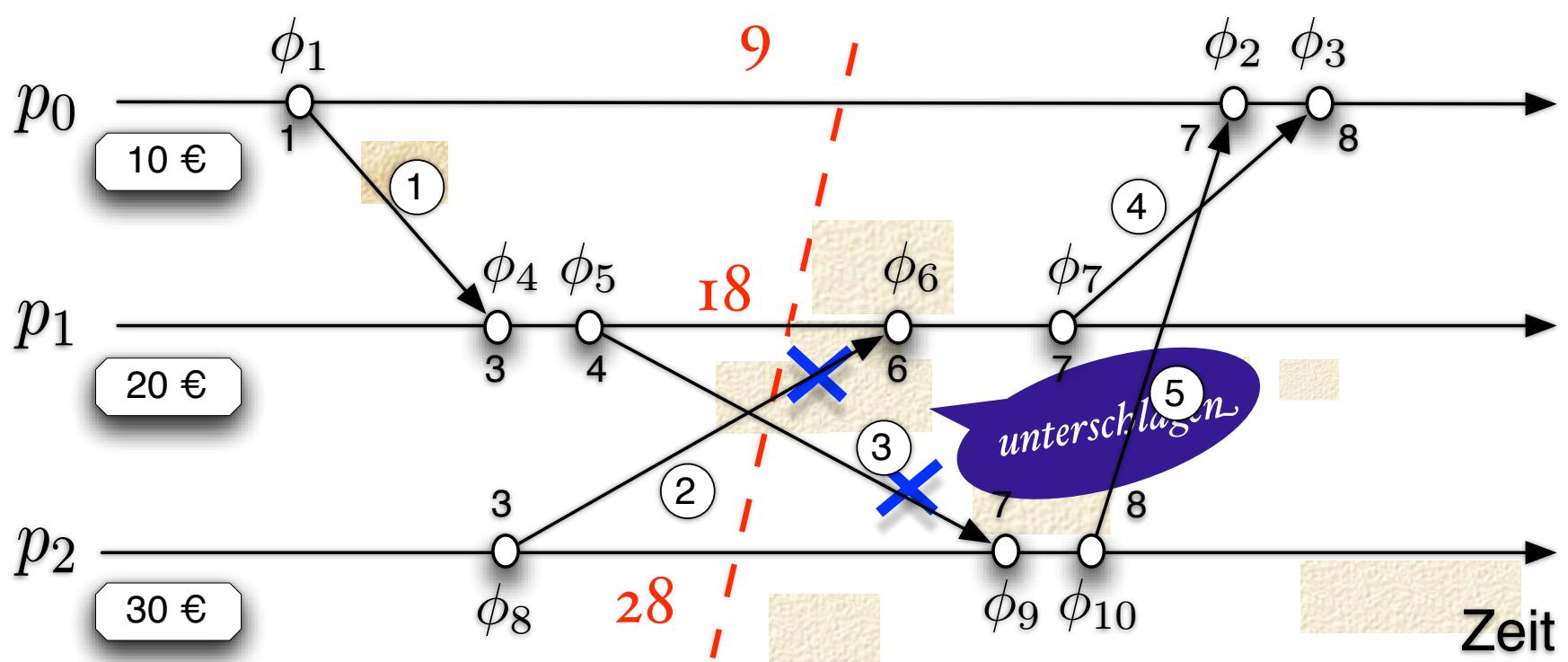
p_2 : 2 an p_1 ;

Beispiel 5.9 Zeitpunkt $t = 5$

unterwegs: 5 \$

also Problem:

die noch nicht empfangenen Nachrichten!



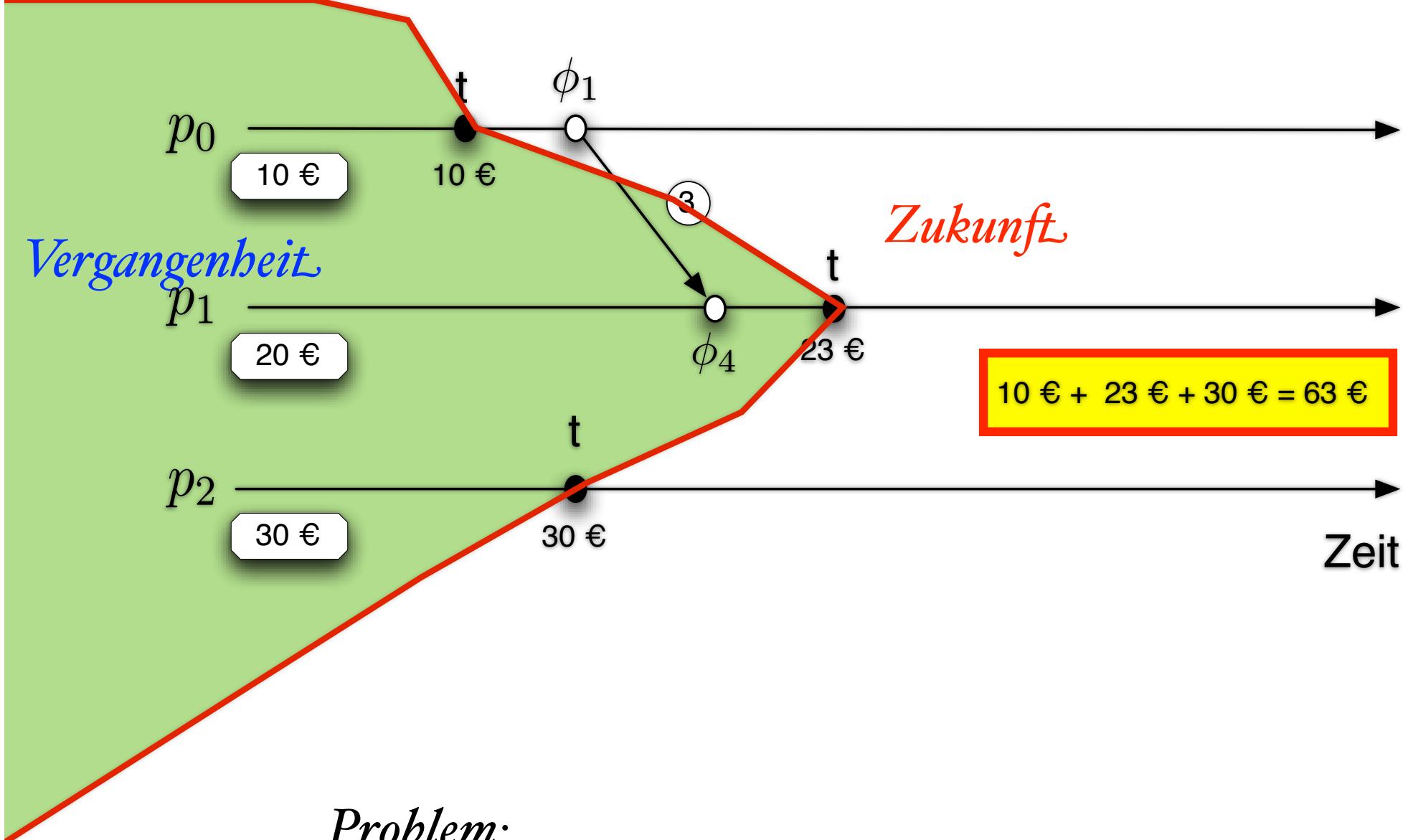
Verfahren 2:

Die Bankleitung bittet die Summe der abgesandten minus der Summe der eingegangenen Beträge zu x_i jeweils hinzuzuzählen.

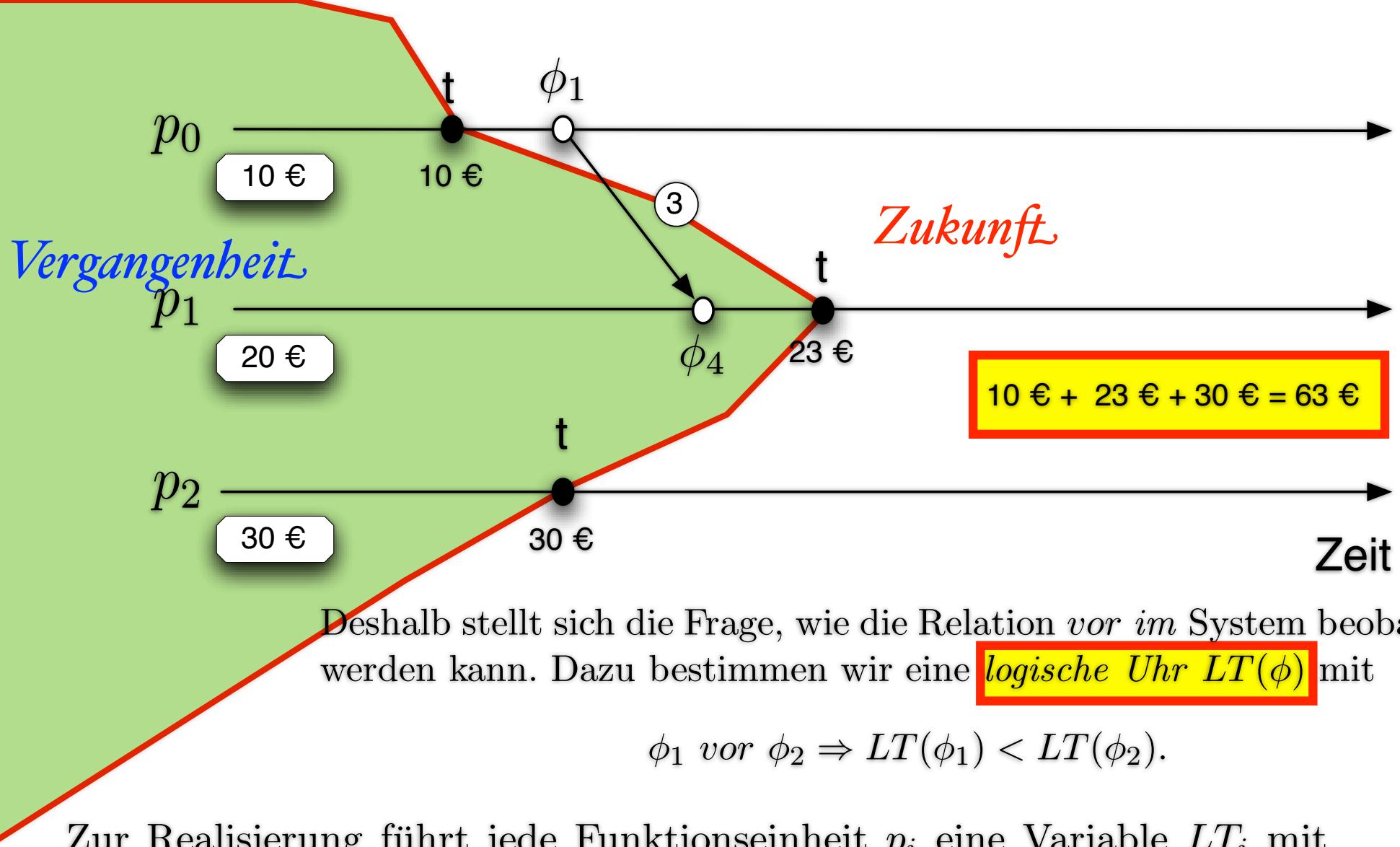
Erwartung : $\sum = 60$.

Beispiel 5.10 Zeitpunkt $t = 5$

$$\begin{array}{rcl}
 p_0 & : & x_0 = 9 + 1 = 10 \\
 p_1 & : & x_1 = 18 - 1 + 3 = 20 \\
 p_2 & : & x_2 = 28 + 2 = 30 \\
 \hline
 & & \sum \quad 60
 \end{array}$$



Problem:
Nachrichten aus der Zukunft in die Vergangenheit !



Zur Realisierung führt jede Funktionseinheit p_i eine Variable LT_i mit Anfangswert $LT_i = 0$ mit. Den Nachrichten wird der neue Wert des Sendeereignisses beigefügt (*logische Zeitstempel*). Ein Ereignis ϕ von p_i setzt LT_i auf einen um 1 größeren Wert als das Maximum des alten Wertes und eines ggf. in ϕ empfangenen Zeitstempels.

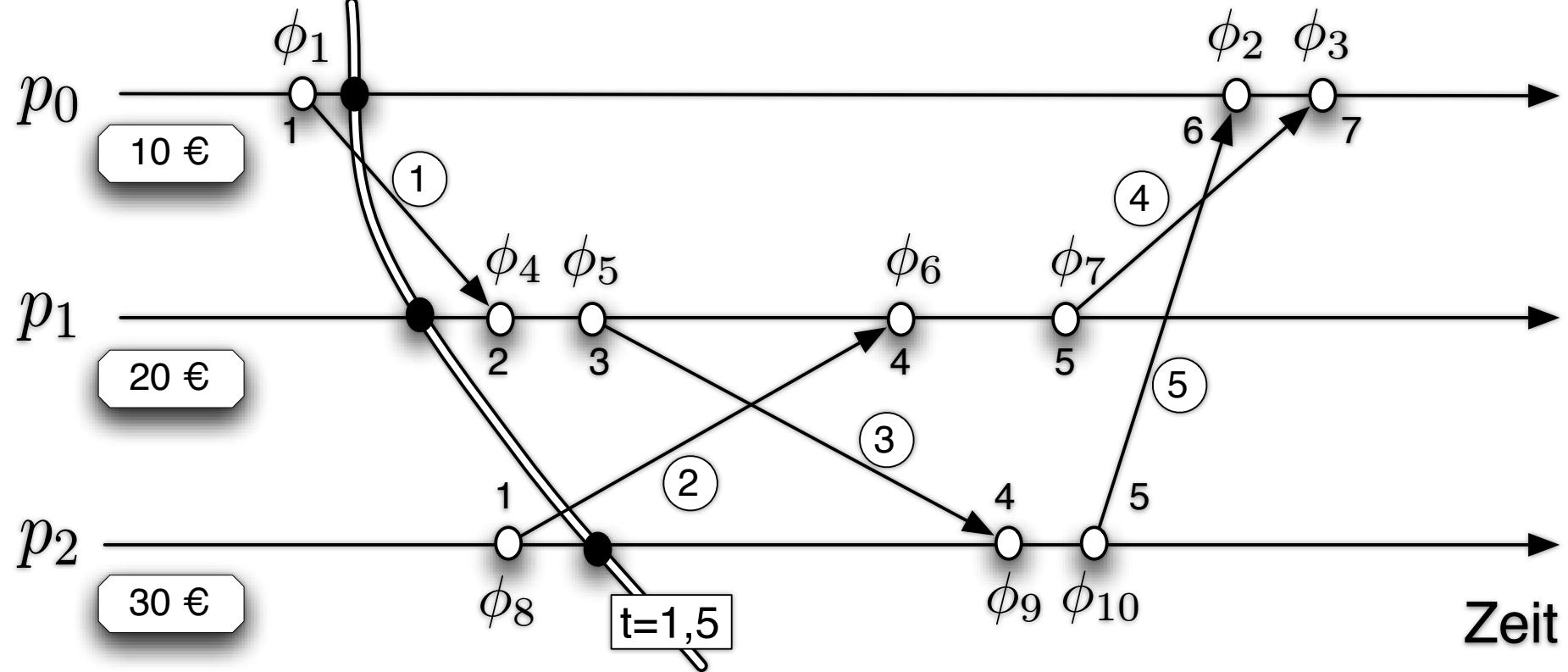


Abbildung 5.9: Zeitskala zu Beispiel 5.13

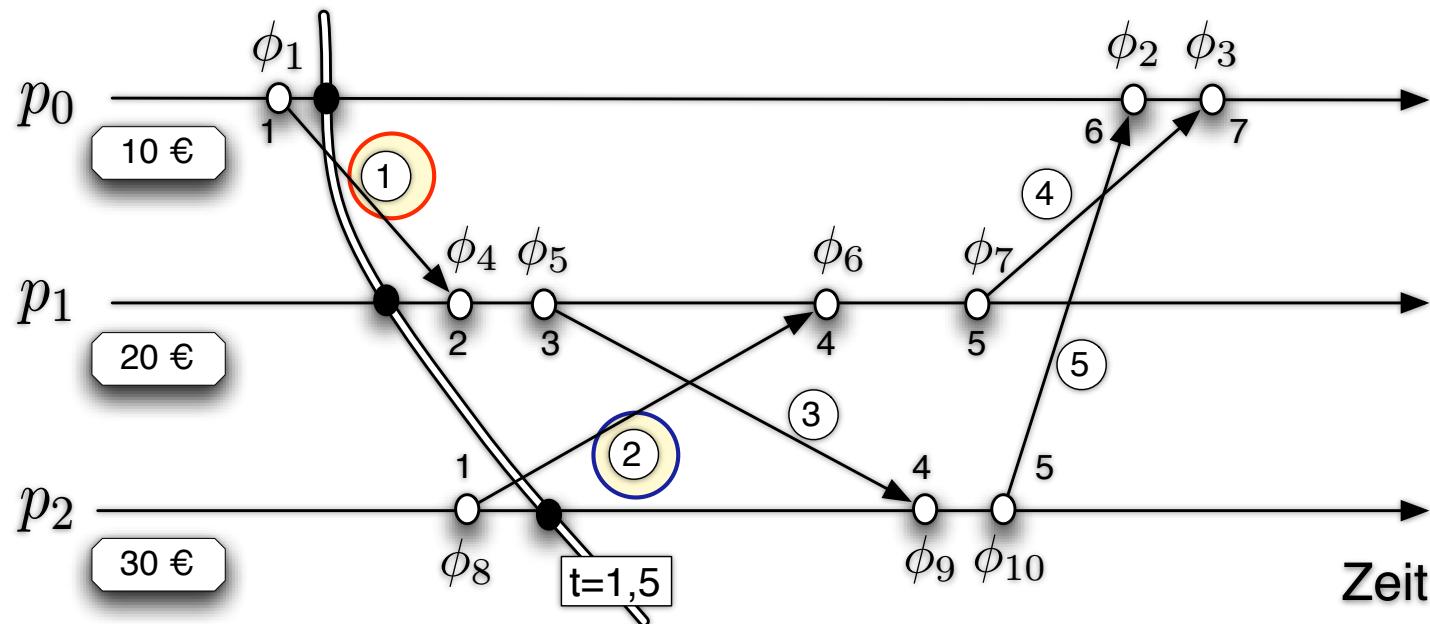
Definition 5.11 Sei ϕ ein Ereignis von p_i . Dann bezeichnet $LT(\phi)$ den von ϕ berechneten Wert von LT_i .

Satz 5.12 Für die Ereignisse $\phi_1, \phi_2 \in \Phi$ gilt:
 ϕ_1 vor $\phi_2 \Rightarrow LT(\phi_1) < LT(\phi_2)$

Algorithmus 5.1 Verfahren der Bankleitung

konsistenter
Schnitt

1. Man führe logische Uhren ein.
2. Man lege ein $t \in \mathbb{Q}$ mit $t \geq 0$ fest.
3. Für jede Funktionseinheit p_i :
 - Bestimme in Bezug auf die lokale Zeit aufeinander folgende Ereignisse ϕ und ϕ' von p_i mit $LT(\phi) \leq t < LT(\phi')$, falls t nicht vor dem ersten Ereignis von p_i liegt.
 - Setze c_i auf den Wert von x_i zwischen ϕ und ϕ' oder auf den Anfangswert, falls t vor dem ersten Ereignis von p_i liegt. Sende c_i an Leitung.
 - Senden den Wert jeder Geldsendung an Leitung, die ab ϕ' ankommt, aber einen Zeitstempel $\leq LT(\phi)$ hat.



Im Beispiel 7.21 sind dies zum Zeitpunkt $t=1,5$:

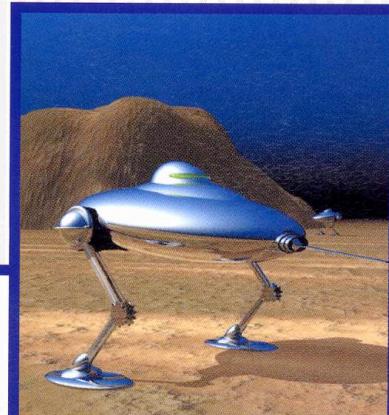
p_0 : eine an p_1

p_1 : keine

p_2 : eine an p_1

Es kommen an : 1 in ϕ_4 und 2 in ϕ_6 .

Die Summe ist $57 + 3 = 60$.



Andrew Tanenbaum
Marten van Steen

Verteilte Systeme

Grundlagen und Paradigmen

Prentice Hall





Jetzt betrachten wir den Algorithmus, den Lamport vorgeschlagen hat, um Ereignissen Zeiten zuzuweisen. Betrachten Sie die drei in Abbildung 5.7 (a) gezeigten Prozesse. Die Prozesse werden auf unterschiedlichen Maschinen ausgeführt, die jeweils eine eigene Uhr haben und mit einer eigenen Geschwindigkeit ausgeführt werden. Wie aus der Abbildung ersichtlich ist, hat die Uhr, wenn sie sechsmal in Prozess 0 getickt hat, in Prozess 1 achtmal getickt, und in Prozess 2 zehnmal. Jede Uhr läuft mit einer konstanten Geschwindigkeit, aber aufgrund von Unterschieden in den Kristallen sind diese Geschwindigkeiten unterschiedlich.

Zum Zeitpunkt 6 sendet der Prozess 0 die Nachricht A an Prozess 1. Wie lange es dauert, bis diese Nachricht ankommt, ist davon abhängig, welcher Uhr Sie glauben. In jedem Fall ist die Uhr in Prozess 1 gleich 16, wenn sie ankommt. Wenn die Nachricht die Startzeit 6 enthält, schließt Prozess 1 daraus, dass die Nachricht für ihren Weg 10 Ticks benötigt hat. Dieser Wert ist sicherlich möglich. Nach dieser Beweisführung benötigt Nachricht B von 1 nach 2 16 Ticks, was ebenfalls ein plausibler Wert ist.

Und jetzt wird es lustig. Nachricht C von 2 an 1 wird zum Zeitpunkt 60 gesendet und kommt zum Zeitpunkt 56 an. Analog dazu wird die Nachricht D von 1 an 0 zum Zeitpunkt 64 gesendet und kommt zum Zeitpunkt 54 an. Diese Werte sind offensichtlich unmöglich. Genau diese Situation muss verhindert werden.

Die von Lamport vorgeschlagene Lösung entsteht direkt aus der Passiert-vor-Relation. Weil C zum Zeitpunkt 60 gesendet wurde, muss sie zum Zeitpunkt 61 oder später ankommen. Aus diesem Grund enthält jede Nachricht die Sendezeit, entsprechend der Uhr des Senders. Kommt eine Nachricht an und die Uhr des Empfängers zeigt einen Wert, der vor der Sendezeit der Nachricht liegt, stellt der Empfänger seine Uhr schnell auf 1 Tick höher als die Sendezeit. In Abbildung 5.7 (b) sehen wir, dass C jetzt zum Zeitpunkt 61 eintrifft, und D analog dazu zum Zeitpunkt 70.

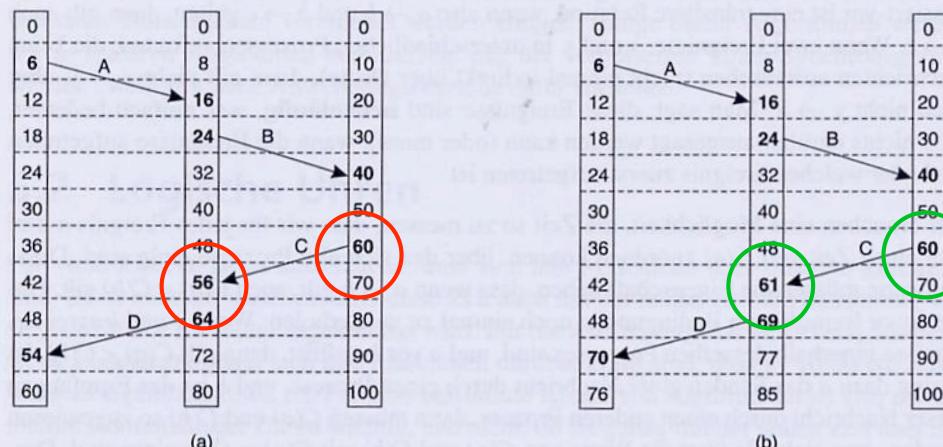


Abbildung 5.7: (a) Drei Prozesse, die jeweils eine eigene Uhr besitzen; die Uhren werden mit unterschiedlichen Geschwindigkeiten ausgeführt. (b) Der Algorithmus von Lamport korrigiert die

Und jetzt wird es lustig. Nachricht C von 2 an 1 wird zum Zeitpunkt 60 gesendet und kommt zum Zeitpunkt 56 an. Analog dazu wird die Nachricht D von 1 an 0 zum Zeitpunkt 64 gesendet und kommt zum Zeitpunkt 54 an. Diese Werte sind offensichtlich unmöglich. Genau diese Situation muss verhindert werden.

Die von Lamport vorgeschlagene Lösung entsteht direkt aus der Passiert-vor-Relation. Weil C zum Zeitpunkt 60 gesendet wurde, muss sie zum Zeitpunkt 61 oder später ankommen. Aus diesem Grund enthält jede Nachricht die Sendezeit, entsprechend der Uhr des Senders. Kommt eine Nachricht an und die Uhr des Empfängers zeigt einen Wert, der vor der Sendezeit der Nachricht liegt, stellt der Empfänger seine Uhr schnell auf 1 Tick höher als die Sendezeit. In Abbildung 5.7 (b) sehen wir, dass C jetzt zum Zeitpunkt 61 eintrifft, und D analog dazu zum Zeitpunkt 70.

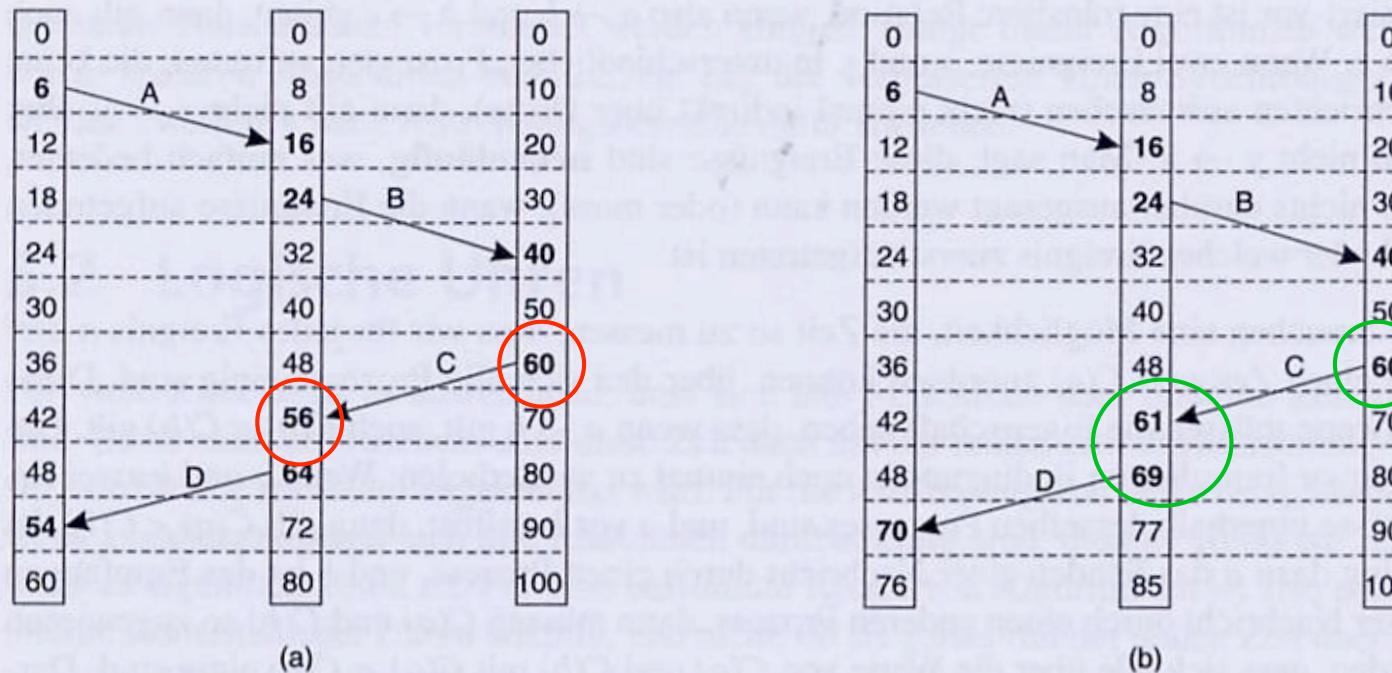


Abbildung 5.7: (a) Drei Prozesse, die jeweils eine eigene Uhr besitzen; die Uhren werden mit unterschiedlichen Geschwindigkeiten ausgeführt. (b) Der Algorithmus von Lamport korrigiert die Uhren.



Beispiel: Vollständig sortiertes Multicasting

Als eine Anwendung der Zeitstempel von Lamport betrachten Sie die Situation, wo eine Datenbank über mehrere Systeme repliziert wurde. Um beispielsweise die Abfrageleistung zu verbessern, kann eine Bank Kopien einer Kontendatenbank in zwei verschiedenen Städten bereitstellen, beispielsweise New York und San Francisco. Eine Abfrage wird immer an die nächstgelegene Kopie weitergegeben. Der Preis für eine schnelle Antwort auf eine Abfrage ist jedoch ein höherer Aktualisierungsaufwand, weil jede Aktualisierungsoperation auf jeder Replik ausgeführt werden muss.

Tatsächlich gibt es eine strengere Forderung in Hinblick auf die Aktualisierungen. Angenommen, ein Kunde in San Francisco will \$ 100 auf sein Konto einzahlen, auf dem sich momentan \$ 1.000 befinden. Gleichzeitig initiiert ein Bankangestellter in New York eine Aktualisierung, bei der dem Konto des Kunden 1 Prozent Zinsen gutgeschrieben werden. Beide Aktualisierungen sollten auf beide Kopien der Datenbank ausgeführt werden. Aufgrund der Kommunikationsverzögerungen im zugrunde liegenden Netzwerk können die Aktualisierungen jedoch in der in Abbildung 5.8 gezeigten Reihenfolge eintreffen.

Die Aktualisierungsoperation des Kunden wird in San Francisco vor der Zinsaktualisierung vorgenommen. Im Gegensatz dazu wird die Kopie des Kontos in der Replik in New York zuerst mit dem einen Prozent Zinsen aktualisiert und danach mit der Einzahlung von \$ 100. Damit enthält die Datenbank in San Francisco einen Gesamtkontostand von \$ 1.111, während die Datenbank in New York nur \$ 1.110 aufweist.



Beispiele

Als eine Datenbank zu Städten immer an einer Ab-

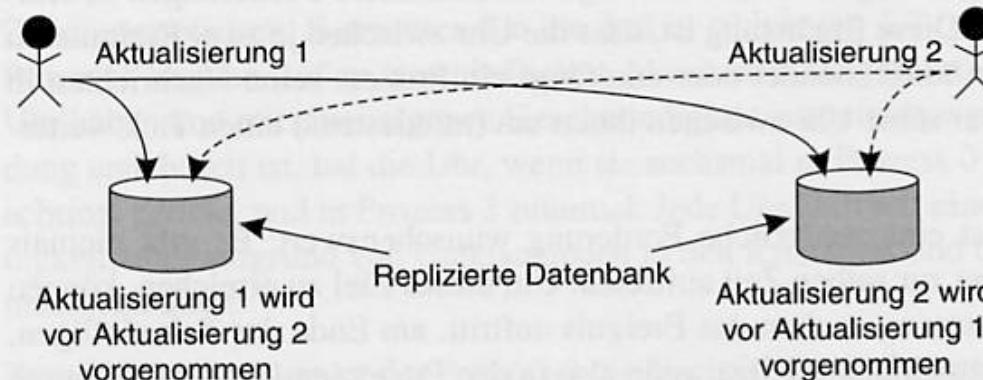


Abbildung 5.8: Aktualisierungen einer replizierten Datenbank und Verursachung eines inkonsistenten Zustands

operation auf jeder Replik ausgeführt werden muss.

Tatsächlich gibt es eine strengere Forderung in Hinblick auf die Aktualisierungen. Angenommen, ein Kunde in San Francisco will \$ 100 auf sein Konto einzahlen, auf dem sich momentan \$ 1.000 befinden. Gleichzeitig initiiert ein Bankangestellter in New York eine Aktualisierung, bei der dem Konto des Kunden 1 Prozent Zinsen gutgeschrieben werden. Beide Aktualisierungen sollten auf beide Kopien der Datenbank ausgeführt werden. Aufgrund der Kommunikationsverzögerungen im zugrunde liegenden Netzwerk können die Aktualisierungen jedoch in der in Abbildung 5.8 gezeigten Reihenfolge eintreffen.

Die Aktualisierungsoperation des Kunden wird in San Francisco vor der Zinsaktualisierung vorgenommen. Im Gegensatz dazu wird die Kopie des Kontos in der Replik in New York zuerst mit dem einen Prozent Zinsen aktualisiert und danach mit der Einzahlung von \$ 100. Damit enthält die Datenbank in San Francisco einen Gesamtkontostand von \$ 1.111, während die Datenbank in New York nur \$ 1.110 aufweist.

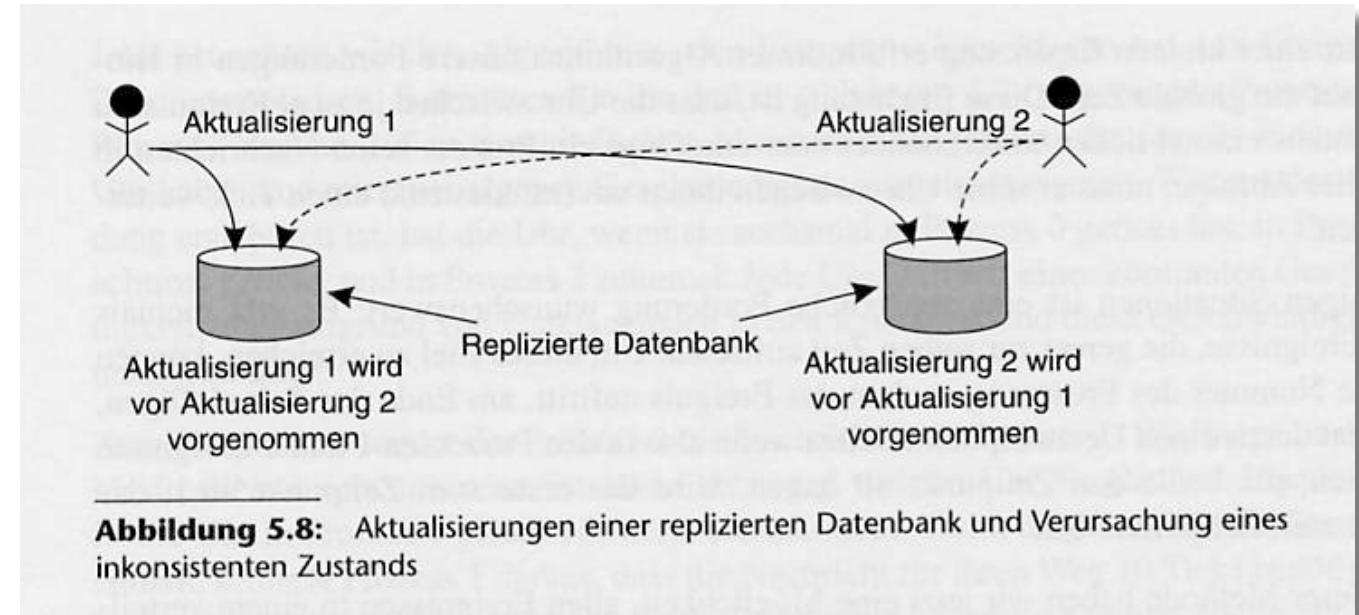
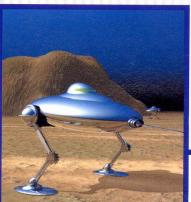


Abbildung 5.8: Aktualisierungen einer replizierten Datenbank und Verursachung eines inkonsistenten Zustands

Der wichtigere Aspekt dabei ist, dass beide Kopien genau gleich sein sollten. Im Allgemeinen benötigt man in solchen Situationen einen **vollständig sortierten Multicast**, d.h. eine Multicast-Operation, wobei alle Nachrichten in derselben Reihenfolge an alle Empfänger ausgeliefert werden. Die Zeitstempel von Lamport können genutzt werden, um diese vollständig sortierten Multicasts auf völlig verteilte Weise zu implementieren.

5.2.2 Vektor-Zeitstempel

Lamport-Zeitstempel führen zu einer Situation, in der alle Ereignisse in einem verteilten System vollständig sortiert sind, mit der Eigenschaft, wenn Ereignis a vor Ereignis b stattgefunden hat, a in dieser Reihenfolge auch vor b positioniert wird, d.h. $C(a) < C(b)$.

Bei Verwendung der Lamport-Zeitstempel kann jedoch nichts über die Beziehung zwischen zwei Ereignissen a und b ausgesagt werden, wenn man nur ihre Zeitwerte $C(a)$ bzw. $C(b)$ vergleicht. Mit anderen Worten, wenn $C(a) < C(b)$ gilt, dann impliziert das nicht unbedingt, dass a tatsächlich vor b stattgefunden hat. Dafür braucht man noch etwas mehr.

Um zu verstehen, was passiert, stellen Sie sich ein Nachrichtensystem vor, wobei die Prozesse Artikel veröffentlichen und auf veröffentlichte Artikel reagieren. Eines der bekannteren Beispiele für ein solches Nachrichtensystem ist das elektronische schwarze Brett im Internet, **Network News** (siehe beispielsweise Comer, 2000b). Benutzer und damit Prozesse treten bestimmten Diskussionsgruppen bei. Die Veröffentlichungen innerhalb einer solchen Gruppe, egal ob es sich dabei um Artikel oder Antworten darauf handelt, werden per Multicast an alle Gruppenmitglieder geschickt. Um sicherzustellen, dass die Reaktionen nach ihren zugehörigen Veröffentlichungen ausgeliefert werden, können wir festlegen, dass ein vollständig sortiertes Multicasting-Schema verwendet werden soll, wie oben beschrieben. Ein solches Schema impliziert jedoch nicht, dass, sollte Nachricht B nach Nachricht A ausgeliefert werden, B tatsächlich eine Reaktion darauf ist, was mithilfe von Nachricht A veröffentlicht wurde. Tatsächlich können die beiden Nachrichten völlig unabhängig voneinander sein. Ein vollständig sortiertes Multicasting ist in diesem Fall zu streng.

Das Problem, das durch die Lamport-Zeitstempel nicht gelöst wird, ist die **Kausalität**. In unserem Beispiel geht der Empfang eines Artikels kausal immer der Veröffentlichung einer Antwort voraus. Müssen also innerhalb einer Gruppe von Prozessen kausale Beziehungen bewahrt werden, sollte der Empfang einer Antwort auf einen Artikel immer dem Empfang dieses Artikels folgen. Nicht mehr und nicht weniger. Wenn zwei Artikel oder Antworten unabhängig voneinander sind, sollte ihre Auslieferungsreihenfolge überhaupt keine Rolle spielen.

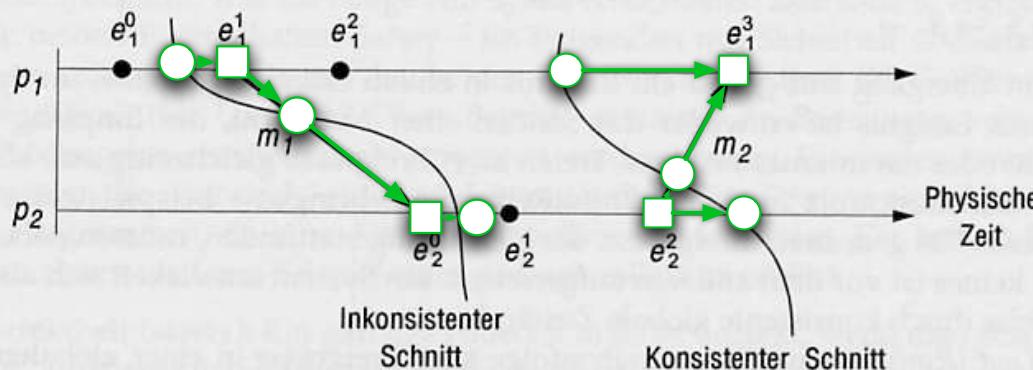
Die Kausalität kann mithilfe von Vektor-Zeitstempeln erfasst werden. Ein Vektor-Zeitstempel $VT(a)$, der einem Ereignis a zugewiesen wurde, hat die Eigenschaft, dass für Er-



ADDISON-WESLEY

Pearson
Studium

Abbildung 10.9 Schnitte

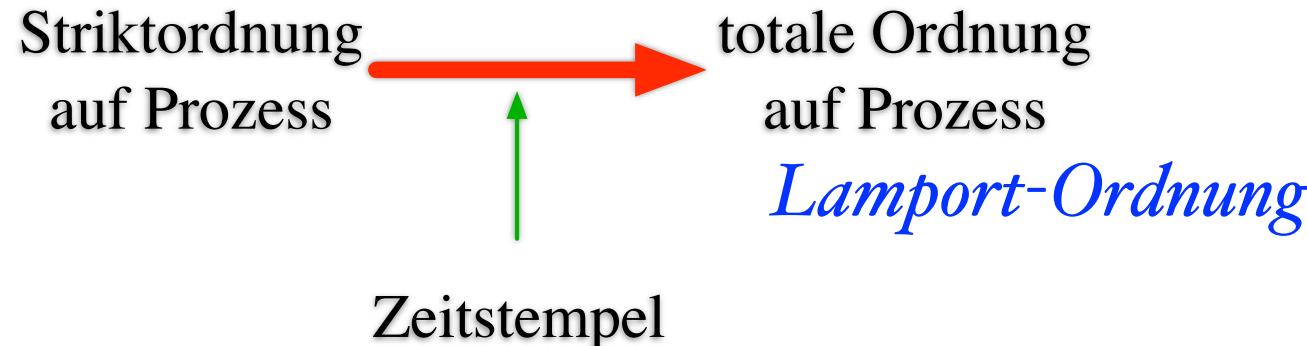


Außerdem können wir die globale History von \wp als die Vereinigung der einzelnen Prozesshistories bilden:

$$H = h_0 \cup h_1 \cup \dots \cup h_{N-1}$$

Mathematisch können wir eine beliebige Menge mit Zuständen der einzelnen Prozesse heranziehen, um einen globalen Zustand $S = (s_1, s_2, \dots, s_N)$ zu bilden. Aber welche globalen Zustände sind dann möglich?

in Anwendungen oft benötigt: totale Ordnung auf Ereignissen im Netz



$$(LT(\phi_1), p_i) < (LT(\phi_2), p_j)$$

gdw.

$$a) LT(\phi_1) < LT(\phi_2) \quad \text{oder}$$

$$b) LT(\phi_1) = LT(\phi_2) \quad \text{und} \quad i < j \quad \text{tie-break-rule}$$

$$(2, p_5) < (3, p_2)$$

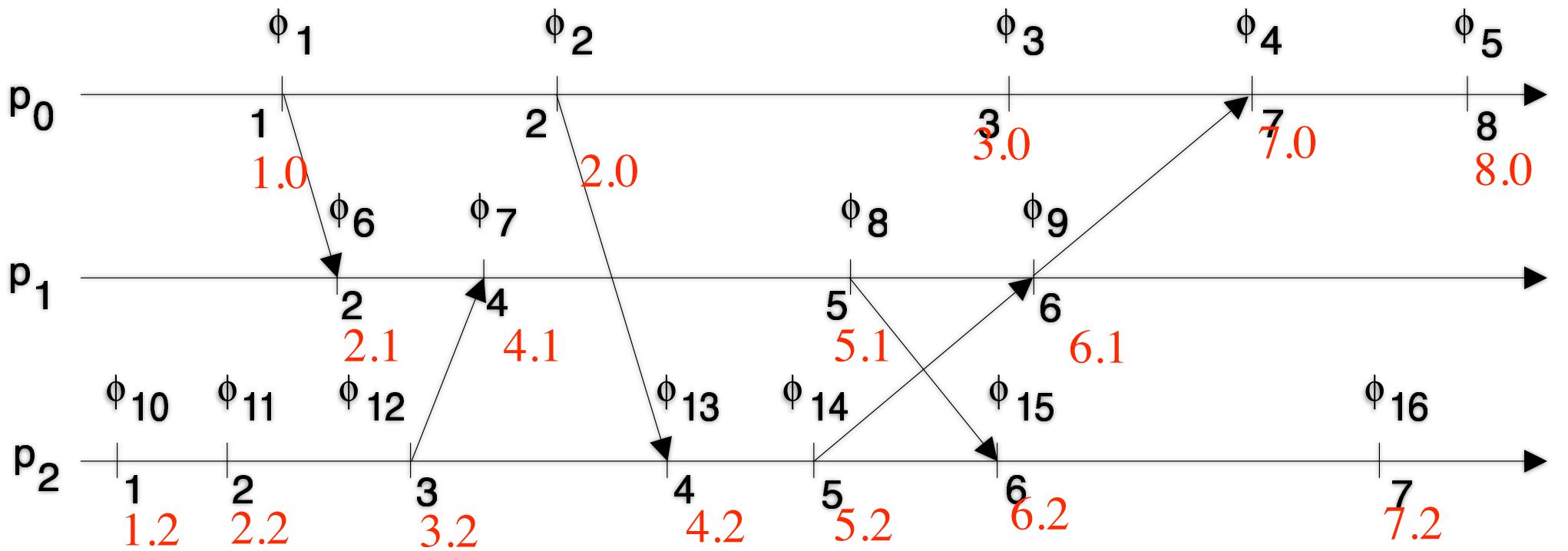
$$(2, p_5) > (2, p_2)$$

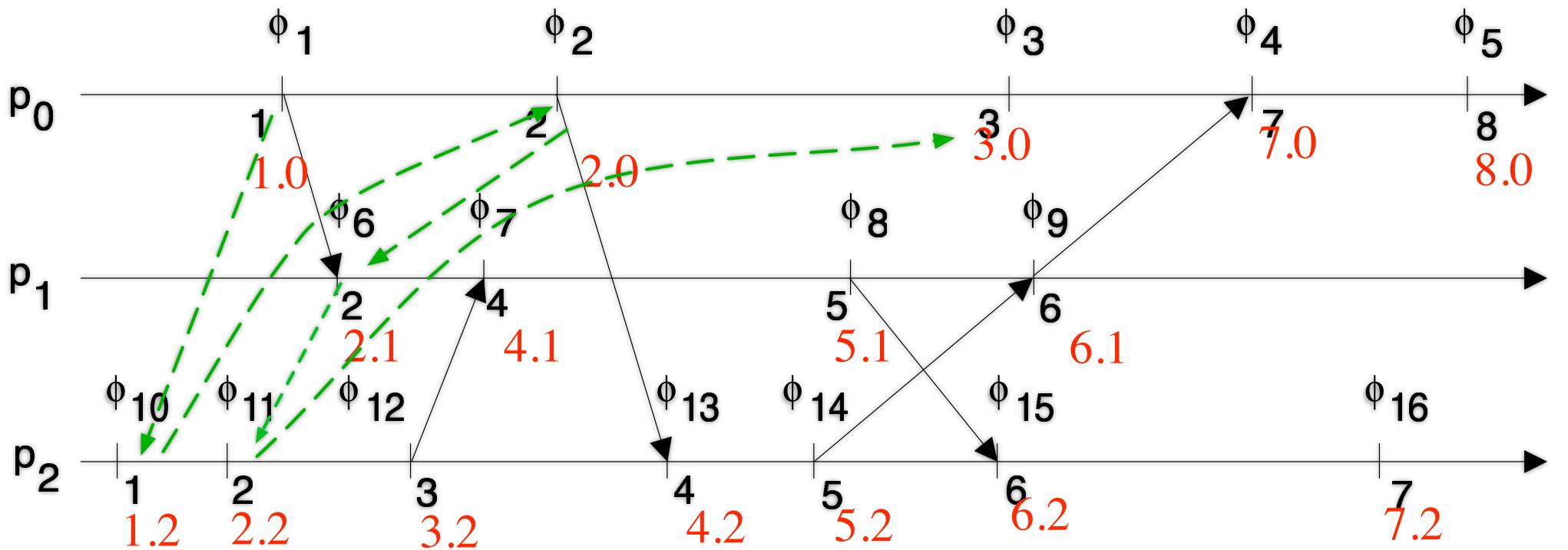
$$(2, p_5) < (4, p_8)$$

L. Lamport

Universität Hamburg * Formale Grundlagen der Informatik 2 *

Lamport-Ordnung





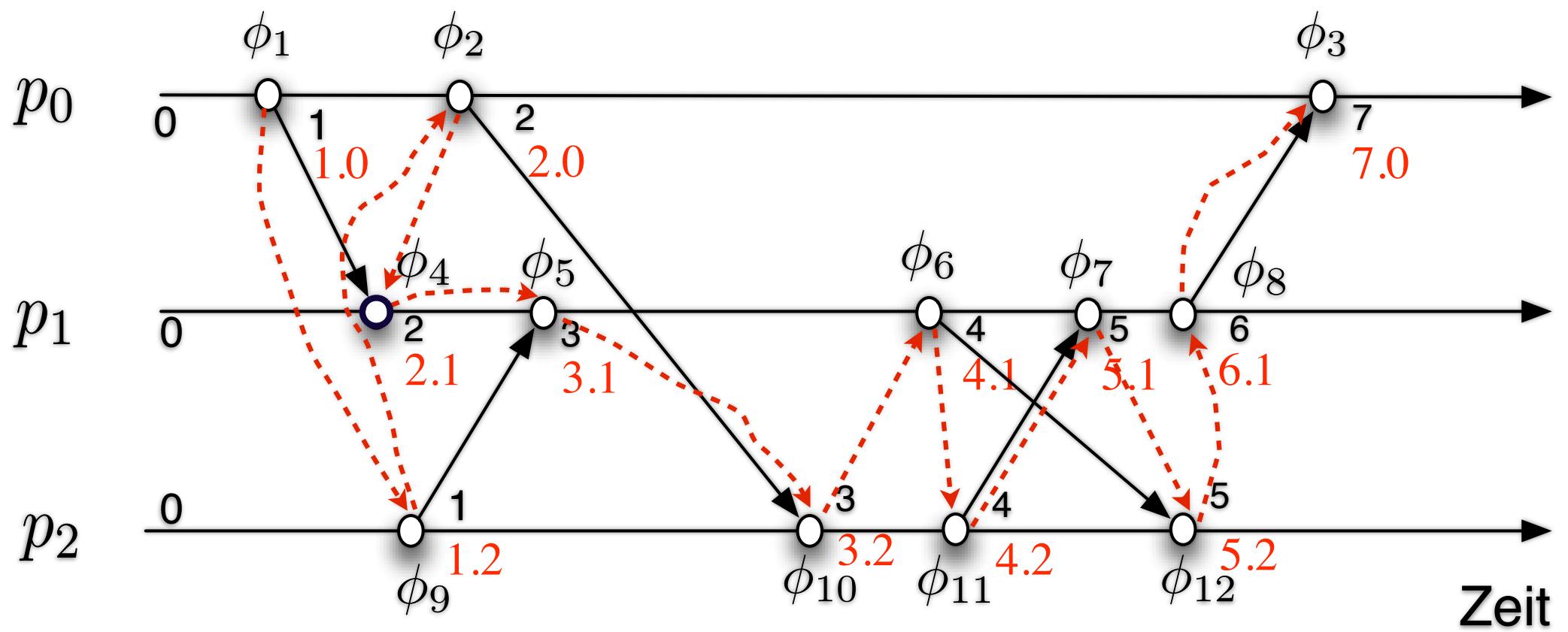


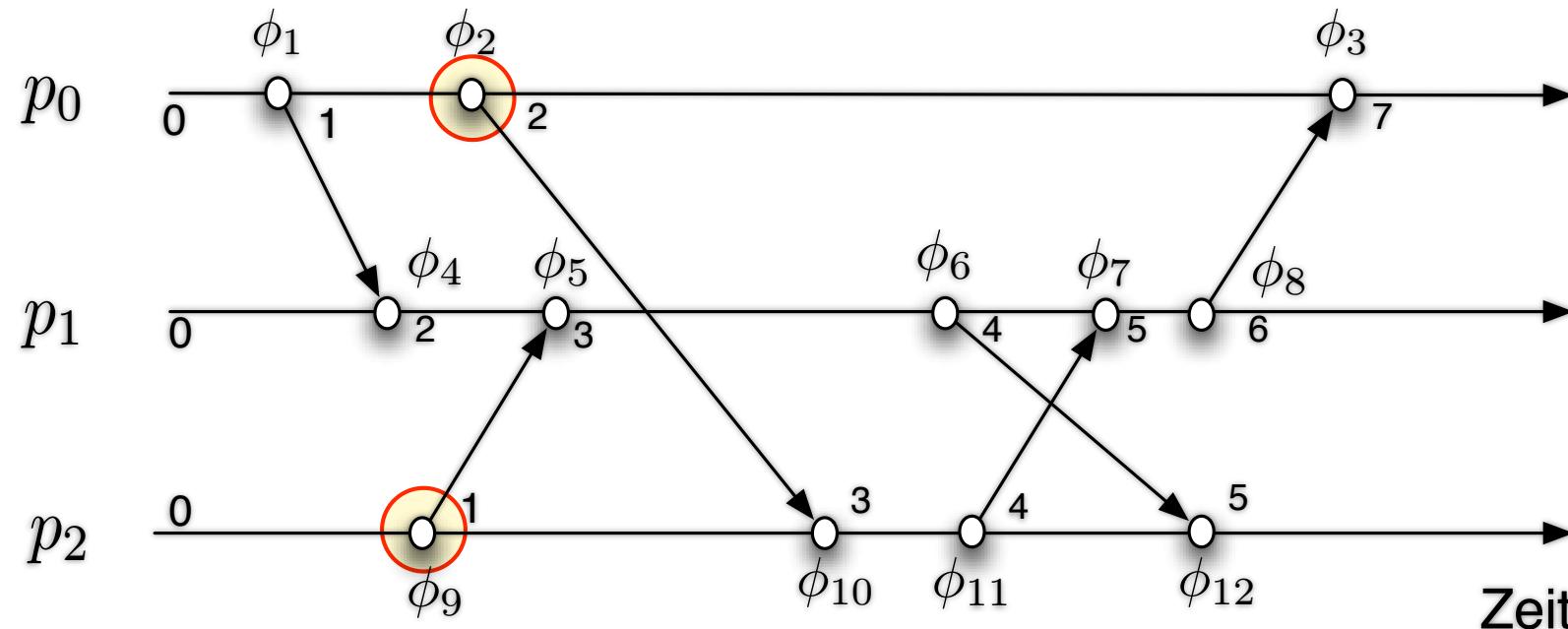
Abbildung 7.19: Senden mit logischen Zeitstempeln

Wir haben gesehen, dass gilt:

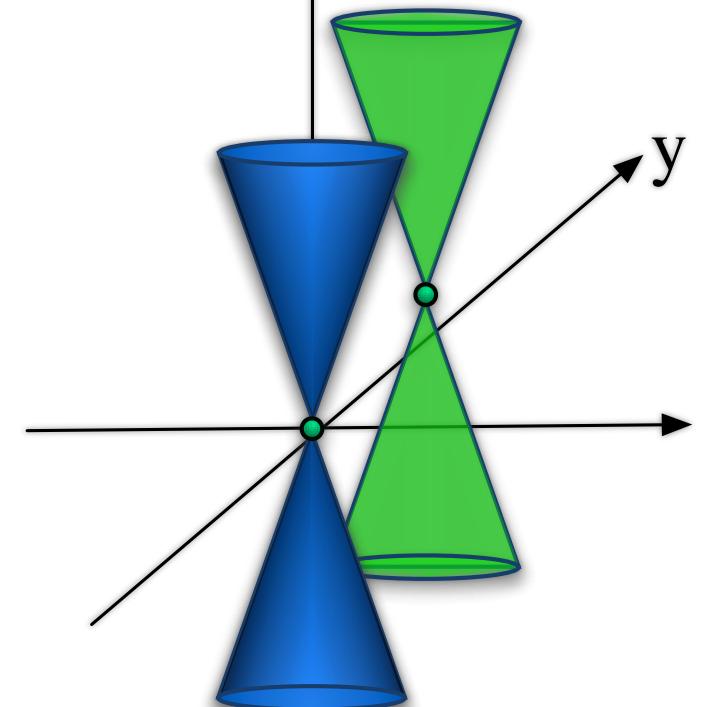
$$\phi_1 \text{ vor } \phi_2 \Rightarrow LT(\phi_1) < LT(\phi_2)$$

Stellt die Relation $<$ die vor-Relation exakt dar, d.h. gilt auch die folgende Umkehrung?

$$LT(\phi_1) < LT(\phi_2) \Rightarrow \phi_1 \text{ vor } \phi_2$$



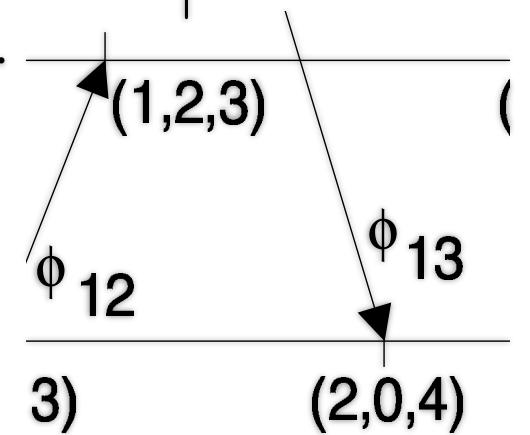
?

$$LT(\phi_1) < LT(\phi_2) \Leftrightarrow \phi_1 \text{ vor } \phi_2$$


Definition 7.24 ϕ_1 heißt **unabhängig** von ϕ_2 bzw. ϕ_1 heißt **nebenläufig** zu ϕ_2 , geschrieben als $\phi_1 \parallel \phi_2$, falls gilt :

$$\phi_1 \parallel \phi_2 \Leftrightarrow \neg(\phi_1 \text{ vor } \phi_2) \wedge \neg(\phi_2 \text{ vor } \phi_1)$$

Gesucht ist eine strikte Ordnung auf Φ , die \parallel darstellt.



Definition 8.22 (Vektorzeit, vektorielle Zeitstempel)

Jede Funktionseinheit p_i führt eine Variable \vec{v}_i mit Werten in \mathbb{N}^n und dem Nullvektor $\vec{0}$ als (Anfangswert) lokale Vektorzeit.

Falls p_i ein Ereignis ϕ bearbeitet, wird der Zeitstempel aktualisiert (vektorieller Zeitstempel):

$$\vec{v}_i[i] \mapsto \vec{v}_i[i] + 1$$

(Inkrementieren des eigenen Stempels)

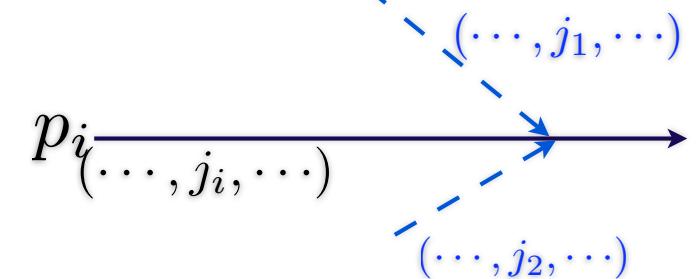
Sei ϕ ein Ereignis von p_i und $\vec{v}_i \in \mathbb{N}^n$ die aktuelle Zeit der Vektoruhr von p_i . Sei ferner

$$\vec{VT}_\phi := \{\vec{VT}_m \mid m \text{ ist empfangene Nachricht in } \phi \text{ mit Stempel } \vec{VT}_m\}.$$

Dann ist für $0 \leq j \leq n - 1$ und $j \neq i$

$$\vec{v}_i[j] \mapsto \max(\{\vec{v}_i[j]\} \cup \vec{VT}_\phi[j])$$

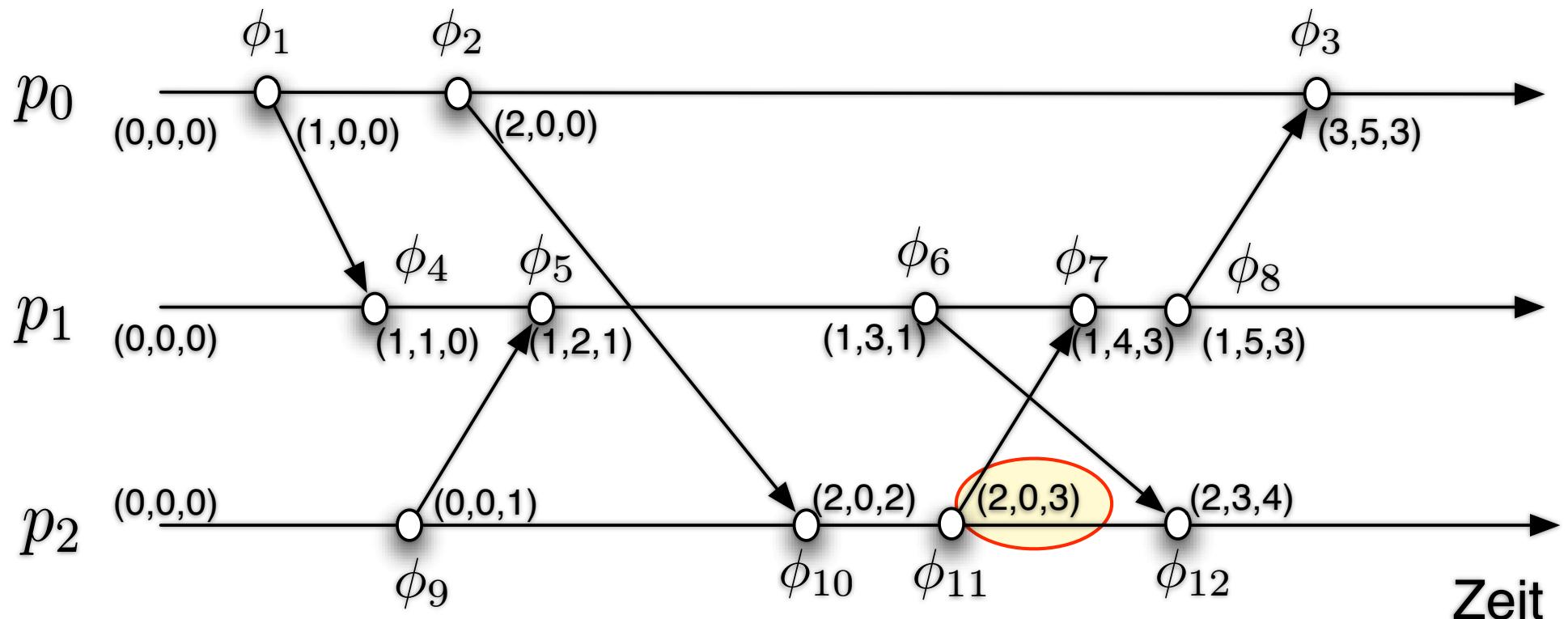
die neue von p_i berechnete Vektorzeit.

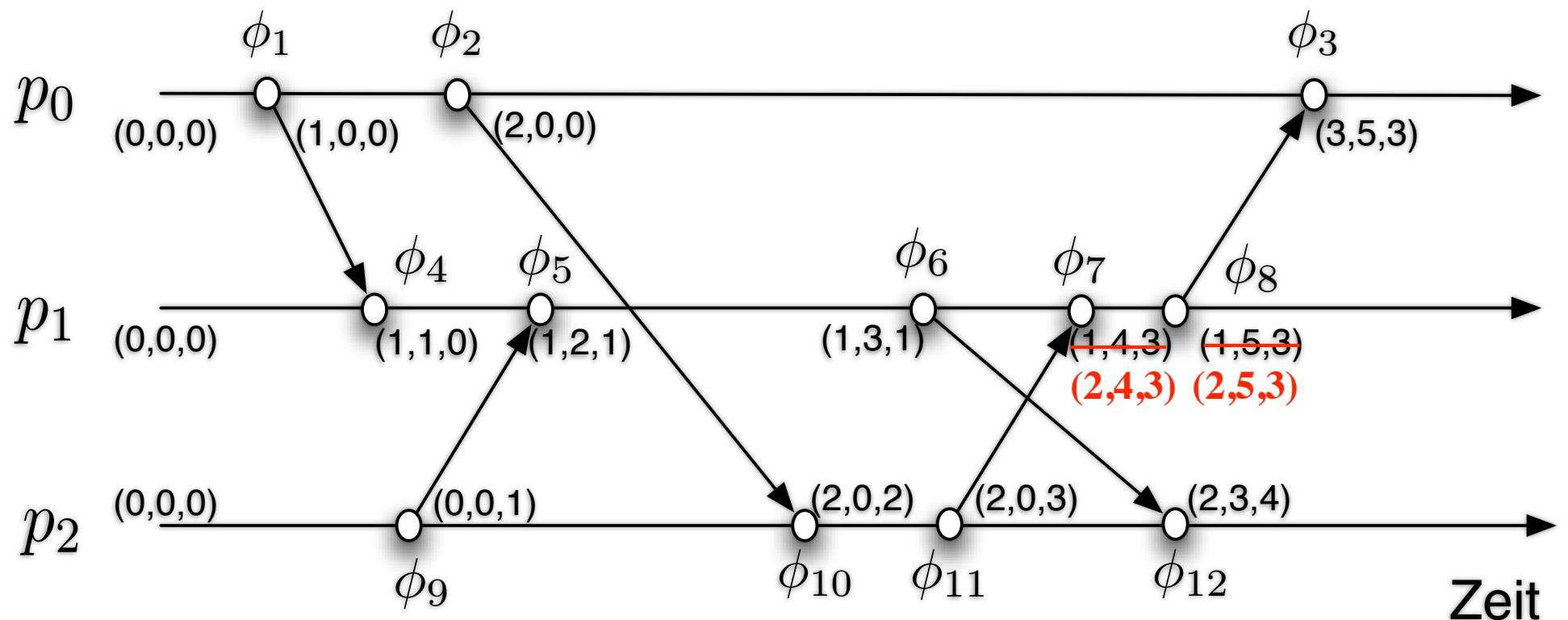


(Aktualisieren der anderen Komponenten)

Definition 7.26 (Vektor-Uhr)

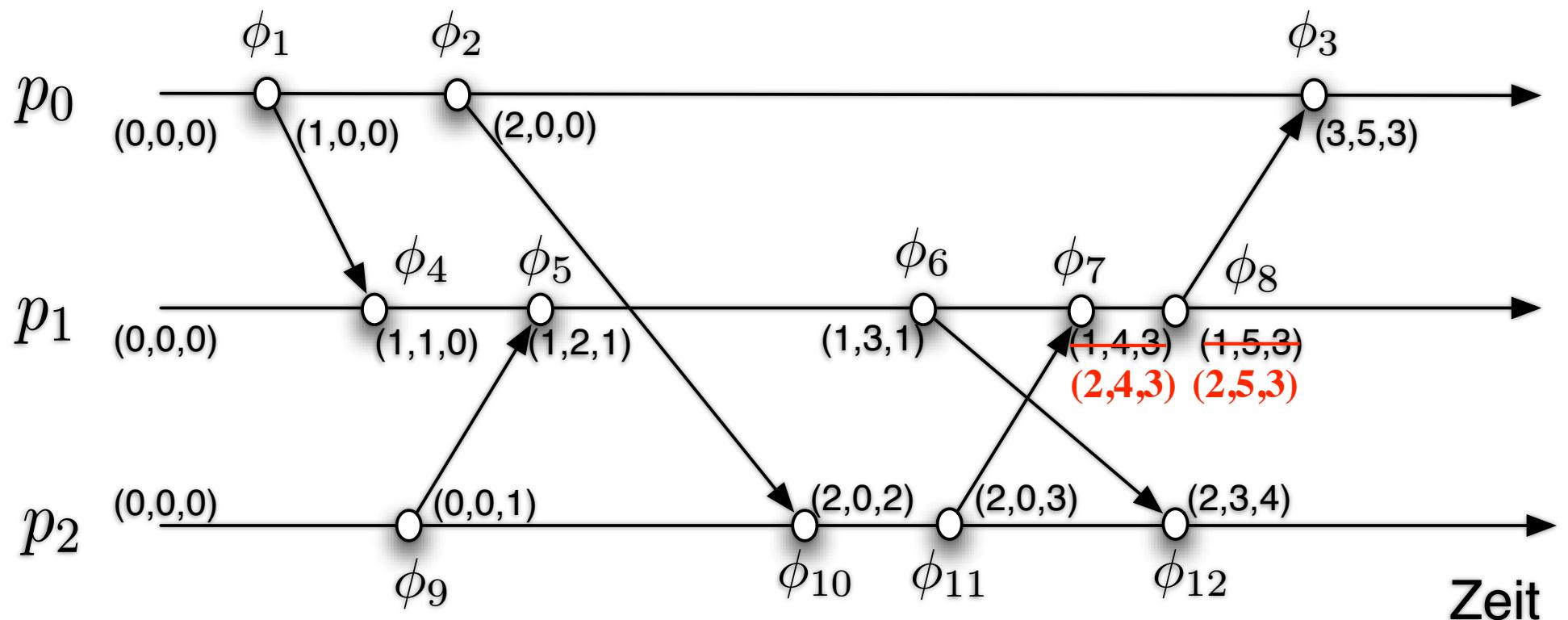
Die Abbildung $VC : \Phi \rightarrow \mathbb{N}^n$ wird definiert durch $VC(\phi) = \vec{v}_i$, wobei \vec{v}_i der von ϕ in p_i berechnete Wert ist.





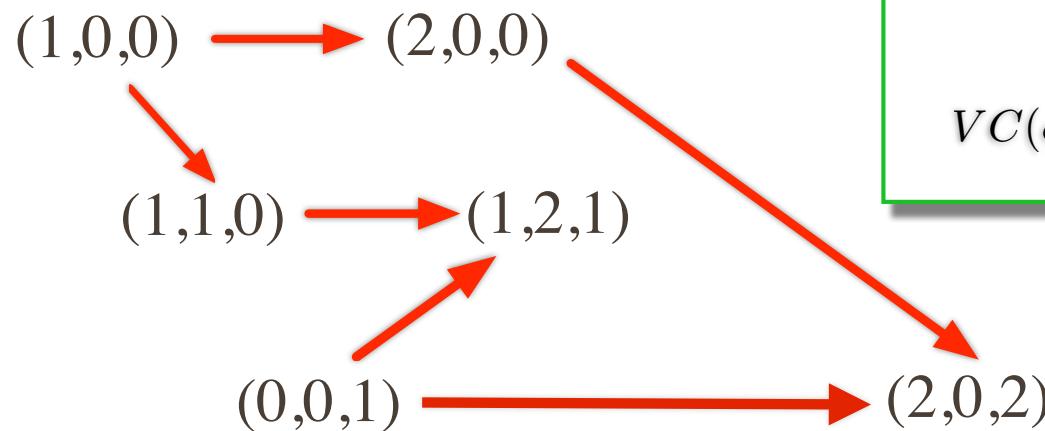
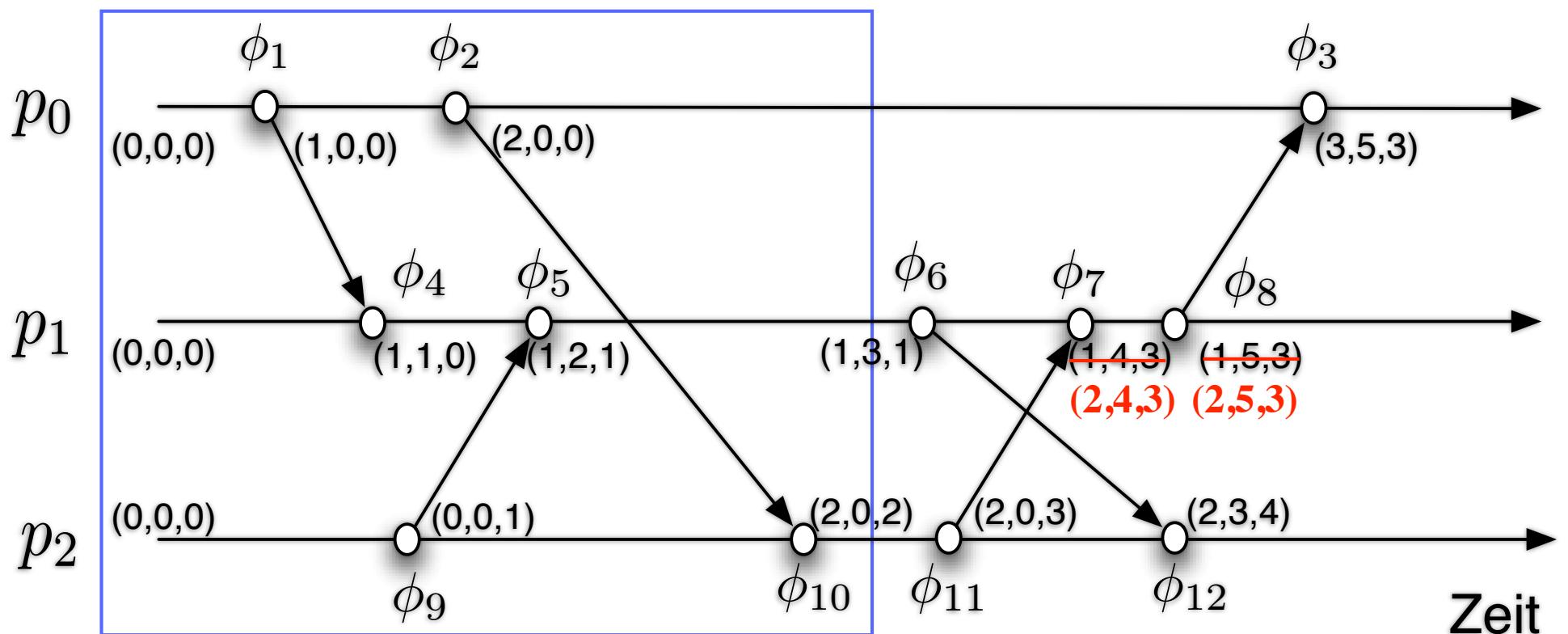
Definition 7.27

- a) Partielle Ordnung auf \mathbb{N}^n : $\vec{v}_1 \leq \vec{v}_2 \Leftrightarrow \forall i \in \{1, \dots, n\} : \vec{v}_1[i] \leq \vec{v}_2[i]$
- b) Strikte Ordnung auf \mathbb{N}^n : $\vec{v}_1 < \vec{v}_2 : \Leftrightarrow \vec{v}_1 \leq \vec{v}_2 \wedge \vec{v}_1 \neq \vec{v}_2$
- c) \vec{v}, \vec{v}' heißen unvergleichbar, falls $\neg(\vec{v} \leq \vec{v}') \wedge \neg(\vec{v}' \leq \vec{v})$ gilt.



Definition 7.24 ϕ_1 heißt **unabhängig** von ϕ_2 bzw. ϕ_1 heißt **nebenläufig** zu ϕ_2 , geschrieben als $\phi_1 \parallel \phi_2$, falls gilt :

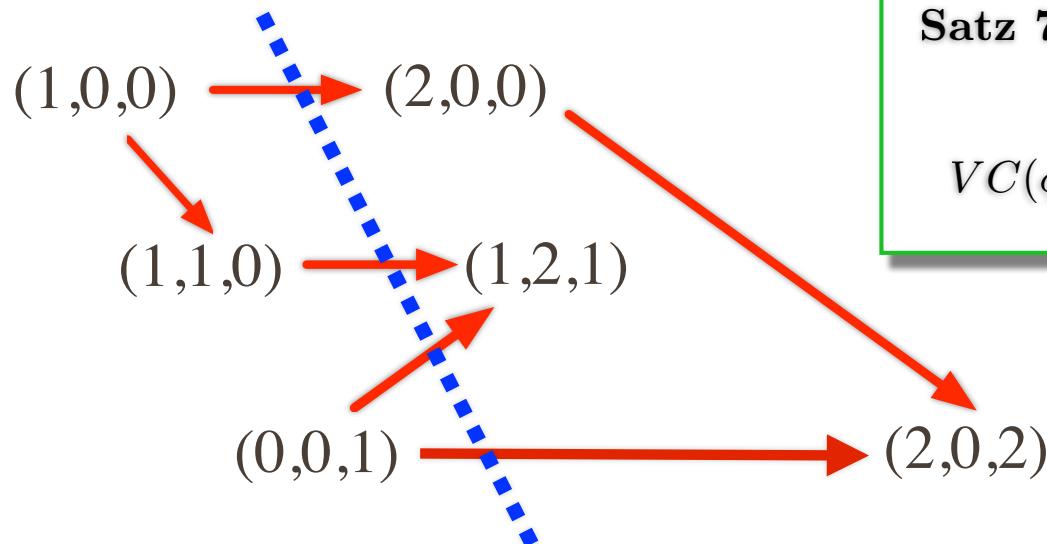
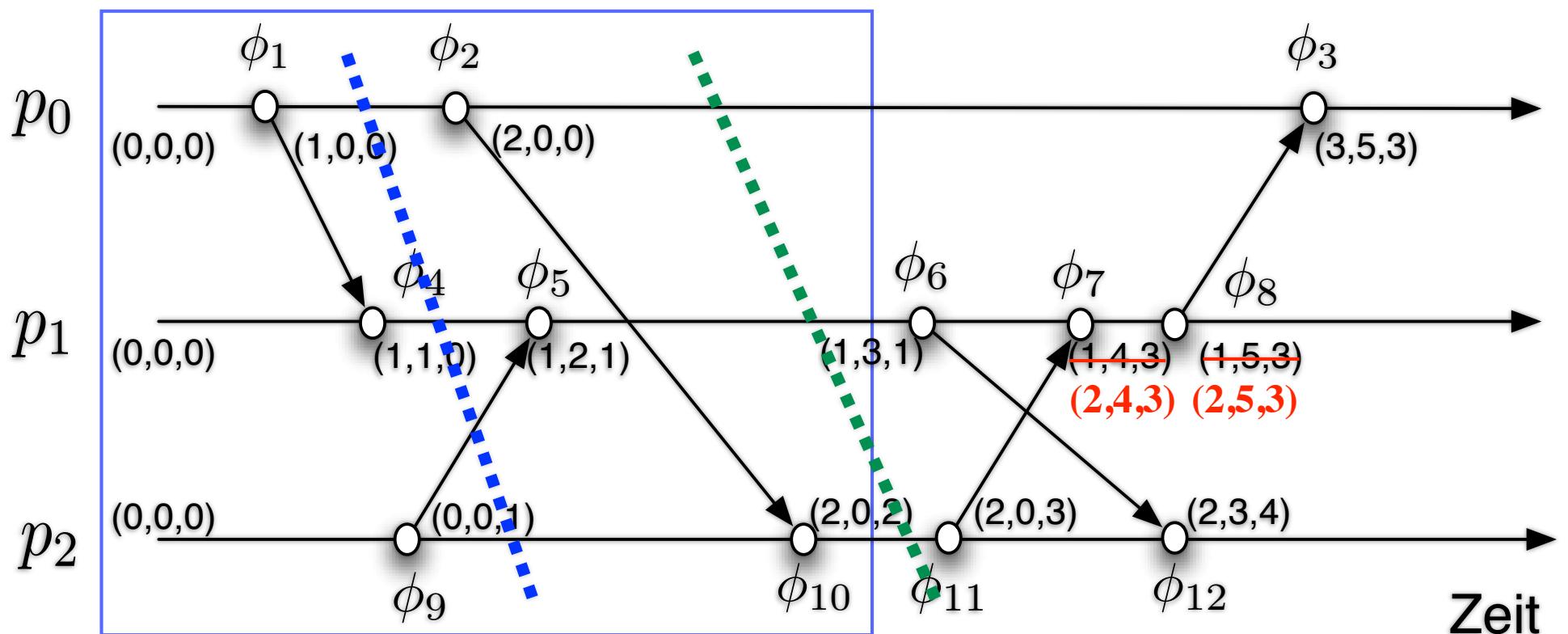
$$\phi_1 \parallel \phi_2 \Leftrightarrow \neg(\phi_1 \text{ vor } \phi_2) \wedge \neg(\phi_2 \text{ vor } \phi_1)$$



Satz 7.28

$VC(\phi_1) < VC(\phi_2) \Leftrightarrow \phi_1 \text{ vor } \phi_2$
 $VC(\phi_1), VC(\phi_2) \text{ unvergleichbar} \Leftrightarrow \phi_1 \parallel \phi_2$

Aufgabe 7.29 Der Bankleitung werden ständig alle Vektor-Zeiten $VC(\phi)$ gesandt. Kann Sie darauf ein Verfahren aufbauen, um das Bilanz-Problem zu lösen?



Satz 7.28

$VC(\phi_1) < VC(\phi_2) \Leftrightarrow \phi_1 \text{ vor } \phi_2$
 $VC(\phi_1), VC(\phi_2) \text{ unvergleichbar} \Leftrightarrow \phi_1 \parallel \phi_2$



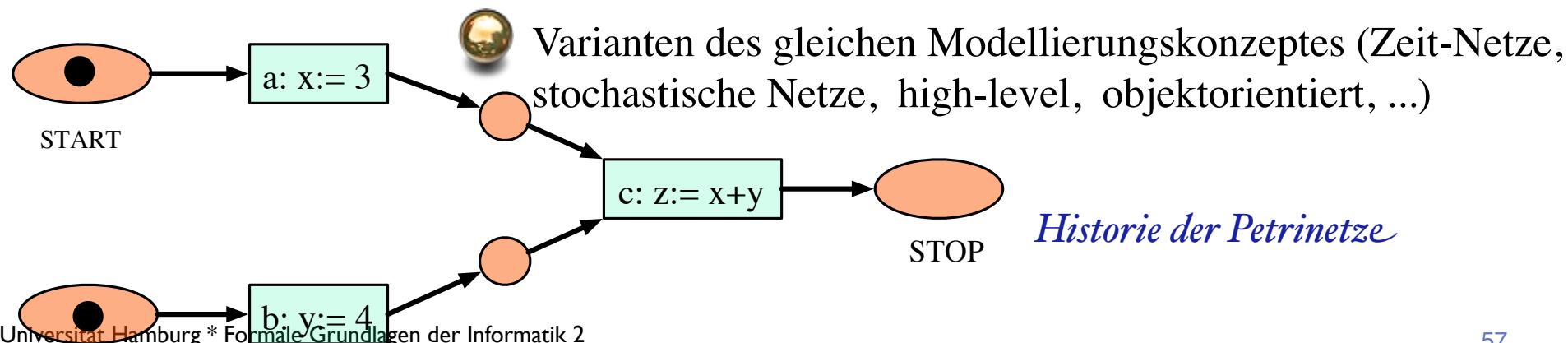
FGI 2

Daniel Moldt

Petrinetze

Allgemeine Eigenschaften von Petrinetzen

- graphische und äquivalente algebraische/textuelle Darstellung
- (formal abgesicherte) Algorithmen für die Analyse
- Abstraktion und hierarchische Strukturen
- hoch entwickelte Theorie der Nebenläufigkeit (concurrency)
- Rechnerwerkzeuge für Editieren, Simulation und Analyse
- siehe z.B. TGI-Tool RENEW <http://www.renew.de/>
- Universalität in Anwendbarkeit (Anwendungen in fast allen Gebieten)



Wer ist Carl Adam Petri?

* 1926 † 2010



<http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/profs/petri.html>

Carl Adam Petri: Dissertation

K o m m u n i k a t i o n
m i t
A u t o m a t e n

Von der Fakultät für Mathematik und Physik
der Technischen Hochschule Darmstadt

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer.nat.)

genehmigte
Dissertation

vorgelegt von
C a r l A d a m P e t r i
aus Leipzig

Referent: Prof.Dr.rer.techn.A.Walther
Korreferent: Prof.Dr.Ing.H.Unger

Tag der Einreichung: 27.7.1961
Tag der mündlichen Prüfung: 20.6.1962

D 17

Bonn 1962

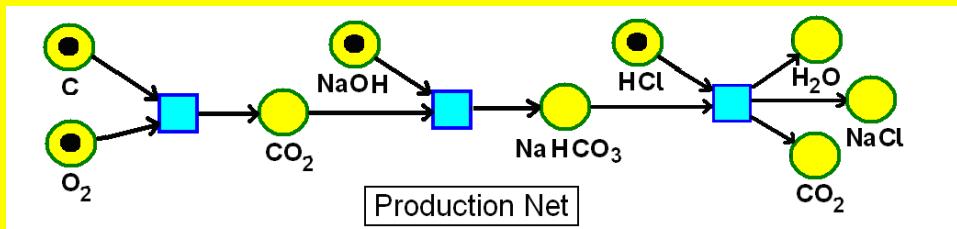
Kommunikation mit Automaten

K o m m u n i k a t i o n
m i t
A u t o m a t e n

Von der Fakultät für Mathematik und Physik

der

The graphics, together with the rules for their coarsening and refinement, were invented in August 1939 by Carl Adam Petri - at the age of 13 - for the purpose of describing chemical processes,

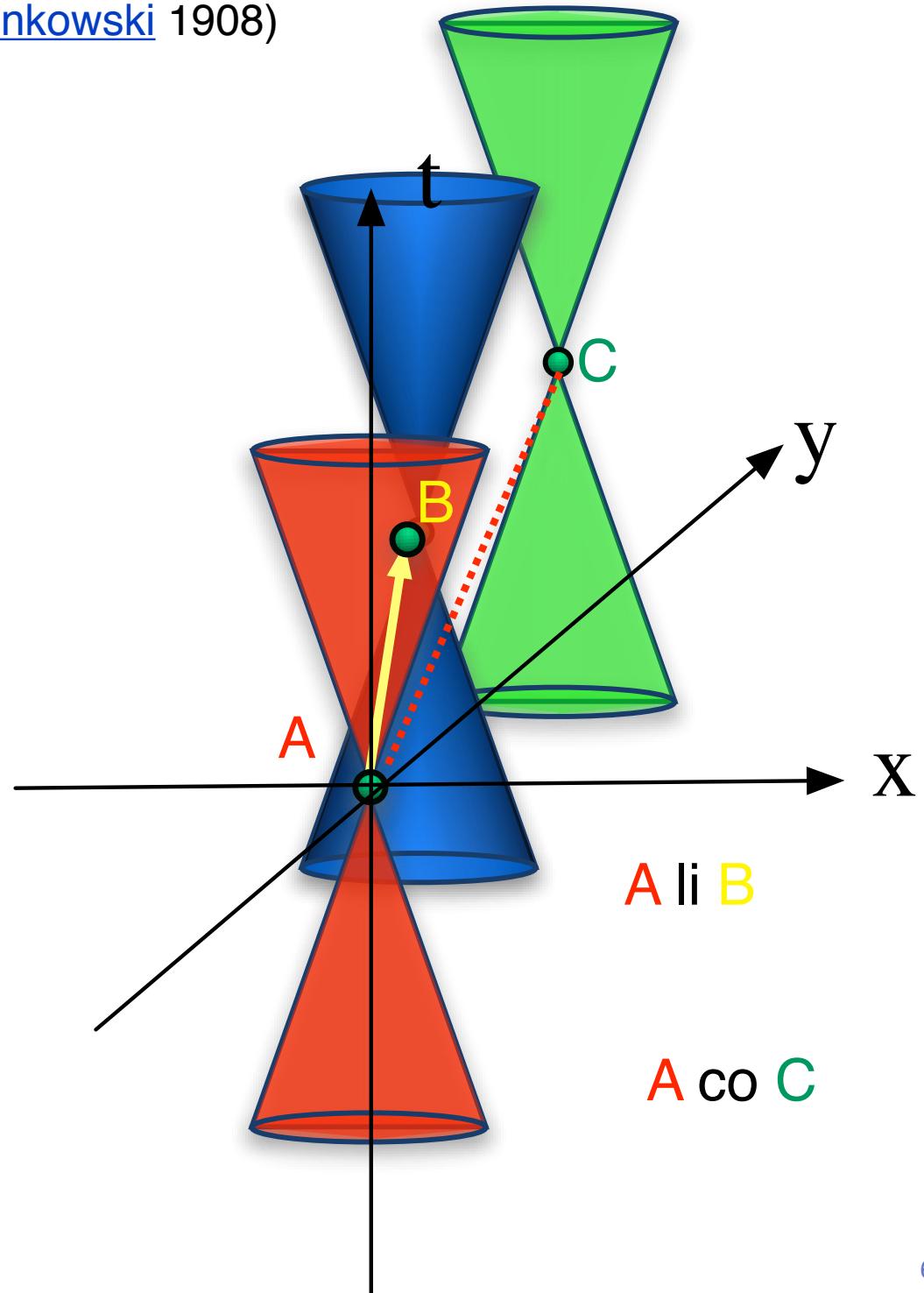
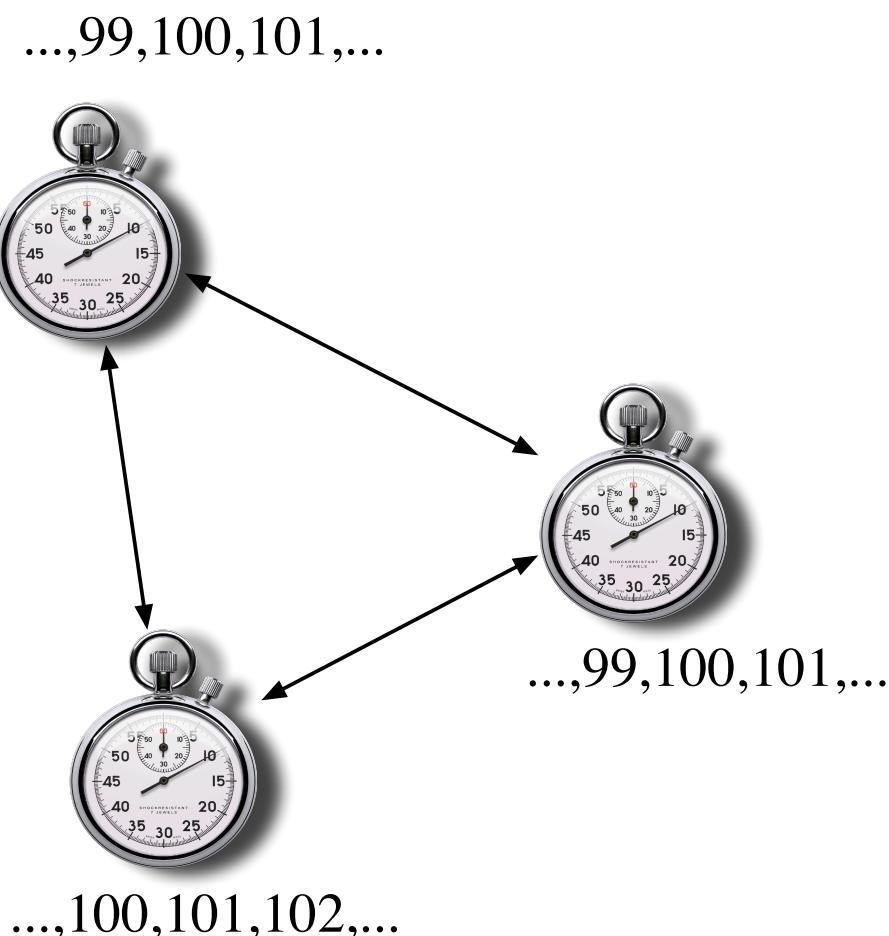


http://www.scholarpedia.org/article/Petri_net

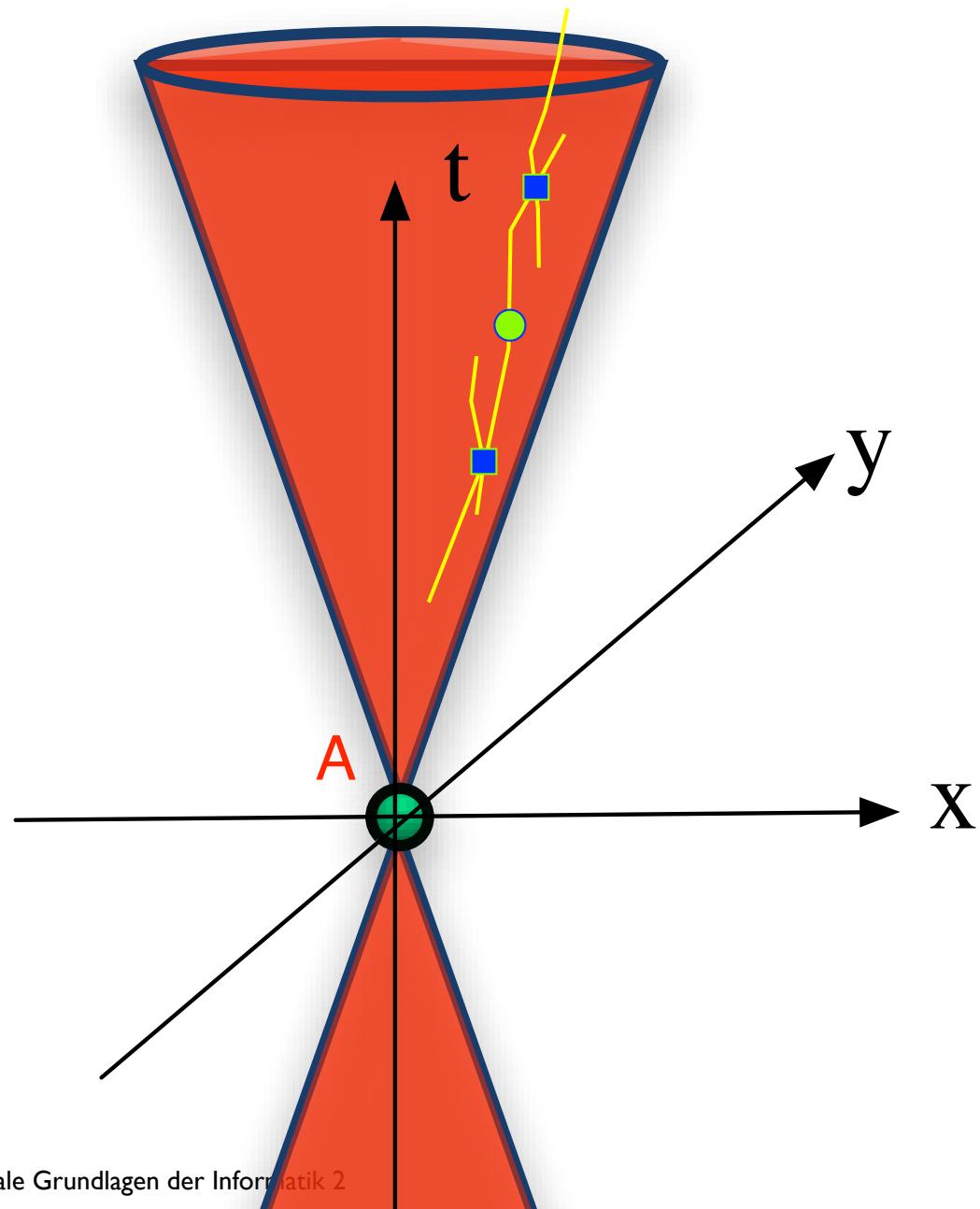
Bonn 1962

Facsimile of the title page of Carl Adam Petri's dissertation

Minkowski-Diagrams ([Hermann Minkowski](#) 1908)



Minkowski-Diagrams (Hermann Minkowski 1908)



Is the Universe a Computer?

Spektrum SPEZIAL
DER WISSENSCHAFT

SPEZIAL 3/07 Spektrum
DER WISSENSCHAFT

FRÜHE IDEEN Konrad Zuses »Rechnender Raum«	MODERNE PHYSIK Das Weltall als Quantenrechner	SPEKULATION Nach uns die Roboter?
--	---	---

Ist das Universum ein Computer?

Von Konrad Zuse und
Carl Friedrich von Weizsäcker
zu Schwarzen Löchern und
bizarren Quantenobjekten

03
4 194058150870
Barcode
€ 8,90 (D) · Österreich € 9,70 · Schweiz SFr 17,40 · Luxemburg € 10,-

www.spektrum.de

PETRI-NETZE

The Universe as a big Net

Das Universum als großes Netz

Die Arbeit von Konrad Zuse wurde in den 1960er Jahren mit der Netztheorie verteilter Systeme weiterentwickelt. Dabei gelang es, einige der Einwände gegen Zuses Konzept auszuräumen.

Von Carl Adam Petri

Wer einen neu konstruierten Computer zum Laufen bringen will, muss eine große Zahl von Einzelteilen zum präzise koordinierten Zusammenspiel bringen – nach dem Vorbild des strengen Wirkens der Naturgesetze. Das hat niemand intensiver erfahren als Konrad Zuse, der Erfinder und Erbauer des weltweit ersten programmgesteuerten Computers.

Der Gedanke liegt nahe, die Naturgesetze direkt für die dauerhafte Stabilität von Computern heranzuziehen. Dann würde der Computer funktionieren, weil er auf Grund der Naturgesetze nicht anders kann. Dazu wäre es freilich notwendig, diese Gesetze in die Sprache des Ingenieurs zu übersetzen. Ist das überhaupt möglich? Wäre es nicht einfacher, die physikalische Natur direkt in der Sprache des Computer-Ingenieurs neu zu be-

schreiben? Wenn das gelänge, erschien die Welt als gigantischer Computer.

Seiner Zeit wieder einmal weit voraus, unternahm Zuse ernsthaft diesen Versuch, zum Kopfschütteln der meisten Zeitgenossen. Er begann mit dem damals bereits bekannten Konzept des »zellulären Automaten«: Hier wird der Raum in lauter gleichartige Zellen aufgeteilt, die wie kleine Maschinen nach einem vorgegebenen Verhaltensmuster (»Programm«) mit ihren Nachbarzellen Information austauschen. Es gelang ihm, wenigstens auf dem Papier, die Phänomene »Fortpflanzung« und »Bewegung« von Zustandsmustern zu beschreiben (siehe seinen Artikel auf S. 6). Für eine Nachprüfung durch Computer-Simulation waren die damaligen Maschinen nicht leistungsfähig genug.

Besser ausgerüstet gelang später anderen die Simulation einfacher zellulärer Automaten mit großem rechnerischem Aufwand. Der zelluläre Automat »Spiel

des Lebens« von John Horton Conway ist weithin bekannt geworden. Vor allem Stephen Wolfram experimentierte mit vielen verschiedenen Zell-Programmen und stellte der Welt 2002 das Ergebnis seiner Versuche mit einem über 1000-seitigen Buch als »Eine Neue Art von Wissenschaft« vor – gewiss ein vollmundiger Titel.

Eines Tages besuchte mich Konrad Zuse in meinem Arbeitszimmer. Ich war ihm empfohlen worden als erforderlicher Theoretiker mit langjähriger Erfahrung als Leiter eines großen Rechenzentrums. Er bat mich, ihn bei seiner Arbeit zum Rechnenden Raum zu beraten. Ich fühlte mich hoch geehrt und stimmte sofort zu. Es entstand eine äußerst fruchtbare Zusammenarbeit, die sich wider Erwartungen über mehr als drei Jahre erstreckte. In ungezählten (Streit-)Gesprächen kamen wir uns wissenschaftlich und persönlich immer näher.

Konrad Zuse vertrat zunächst seinen Ansatz der zellulären Automaten und verteidigte ihn vehement gegen meine Bedenken. Diese bestanden hauptsächlich in Folgendem: Erstens ergibt sich aus der räumlichen Anordnung der Zellen eine Auszeichnung von drei bestimmten Richtungen im Raum, die sich nur durch Einführung zufälliger Prozesse aufheben lässt. (Zuse hielt die Annahme von Zufälligkeit nicht für zielführend.)

Zweitens wandte ich ein, dass sich in diesem Modell die gesamte Physik in den Programmen jeder Zelle verstecke, sozusagen als nicht analysierbare DNA und ohne direkten Bezug zu physikalischen Größen. Ich schlug ihm deshalb eine andere Modellierungstechnik vor: die Netztheorie verteilter Systeme. Heute unter dem Namen »Petri-Netze« bekannt, hatte sie bereits viele erfolgreiche Anwendungen gefunden, so in Bankwesen, Ökonomie, Telekommunikation, Workflow Management, Konfliktlösung, Prozesssteuerung und Biochemie, nur leider noch nicht an ihrem Geburtsort,



Carl Adam Petri (links) im Gespräch mit Konrad Zuse, um 1975

The Universe as a big Net

Das Universum als großes Netz

Die Arbeit von Konrad Zuse wurde in den 1960er Jahren mit der Netztheorie verteilter Systeme weiterentwickelt. Dabei gelang es, einige der Einwände gegen Zuses Konzept auszuräumen.

Von Carl Adam Petri

Wer einen neu konstruierten Computer zum Laufen bringen will, muss eine große Zahl von Einzelteilen zum präzise koordinierten Zusammenspiel bringen – nach dem Vorbild des strengen Wirkens der Naturgesetze. Das hat niemand intensiver erfahren als Konrad Zuse, der Erfinder und Erbauer des weltweit ersten programmgesteuerten Computers.

Der Gedanke liegt nahe, die Naturgesetze direkt für die dauerhafte Stabilität von Computern heranzuziehen. Dann würde der Computer funktionieren, weil er auf Grund der Naturgesetze nicht anders kann. Dazu wäre es freilich notwendig, diese Gesetze in die Sprache des Ingenieurs zu übersetzen. Ist das überhaupt möglich? Wäre es nicht einfacher, die

schreiben? Wenn das gelänge, erschien die Welt als gigantischer Computer.

Seiner Zeit wieder einmal weit voraus, unternahm Zuse ernsthaft diesen Versuch, zum Kopfschütteln der meisten Zeitgenossen. Er begann mit dem damals bereits bekannten Konzept des »zellulären Automaten«: Hier wird der Raum in lauter gleichartige Zellen aufgeteilt, die wie kleine Maschinen nach einem vorgegebenen Verhaltensmuster (»Programm«) mit ihren Nachbarzellen Information austauschen. Es gelang ihm, wenigstens auf dem Papier, die Phänomene »Fortpflanzung« und »Bewegung« von Zustandsmustern zu beschreiben (siehe seinen Artikel auf S. 6). Für eine Nachprüfung durch Computer-Simulation waren die damaligen Maschinen nicht leistungsfähig genug.

Besser ausgerüstet gelang später anderer die Simulation einfacher zellulärer

des Lebens« von John Horton Conway ist weithin bekannt geworden. Vor allem Stephen Wolfram experimentierte mit vielen verschiedenen Zell-Programmen und stellte der Welt 2002 das Ergebnis seiner Versuche mit einem über 1000-seitigen Buch als »Eine Neue Art von Wissenschaft« vor – gewiss ein vollmundiger Titel.

Eines Tages besuchte mich Konrad Zuse in meinem Arbeitszimmer. Ich war ihm empfohlen worden als erforderlicher Theoretiker mit langjähriger Erfahrung als Leiter eines großen Rechenzentrums. Er bat mich, ihn bei seiner Arbeit zum Rechnenden Raum zu beraten. Ich fühlte mich hoch geehrt und stimmte sofort zu. Es entstand eine äußerst fruchtbare Zusammenarbeit, die sich wider Erwartungen über mehr als drei Jahre erstreckte. In ungezählten (Streit-)Gesprächen kan-

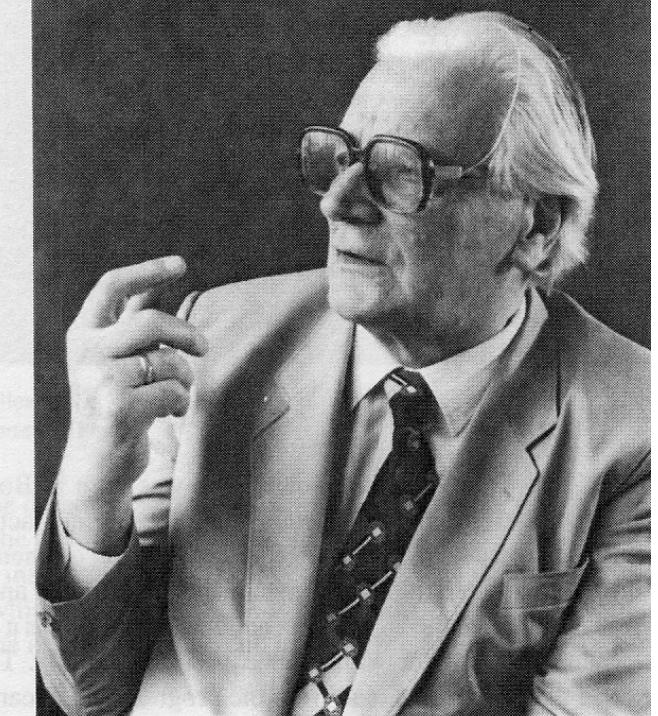


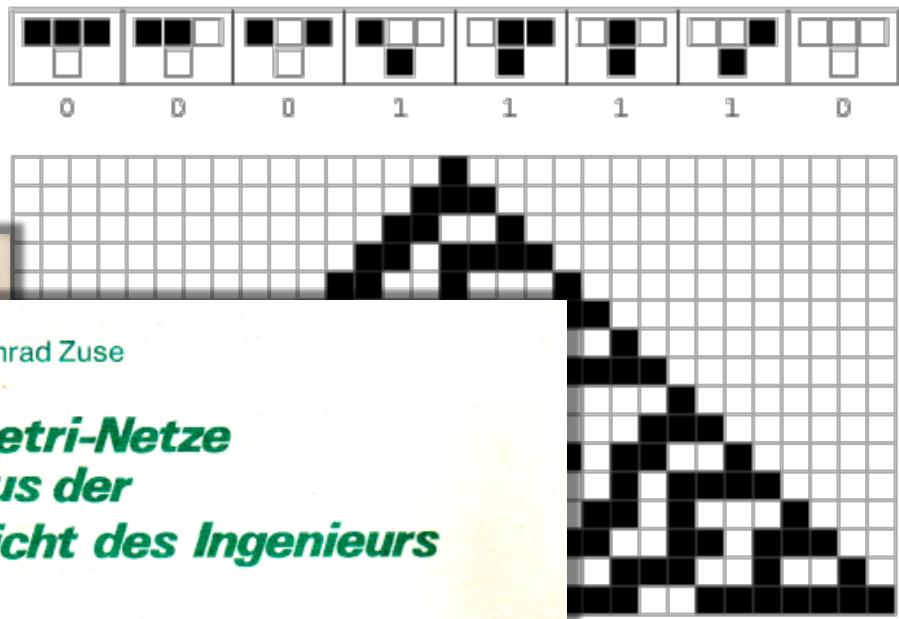
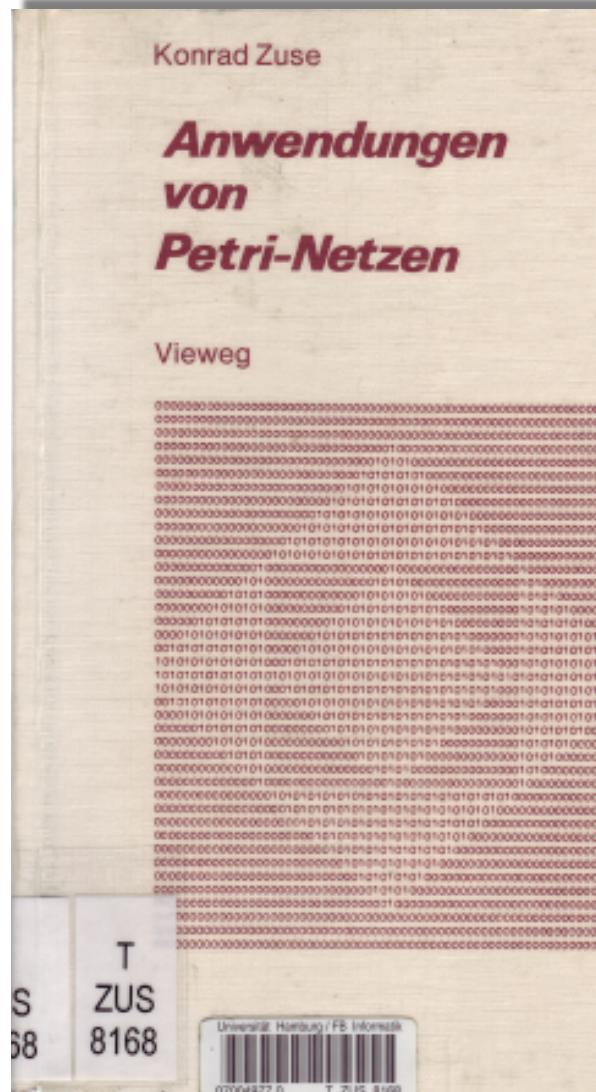
Fig. 2.28 Konrad Zuse
Courtesy of Horst Zuse, Berlin.

Today, in the whole world Konrad Zuse is accepted as the creator / inventor of the first free programmable computer with a binary floating point and switching system, which really worked.

This machine - called Z3 - was completed in his small workshop in Berlin in 1941. First thoughts of Konrad Zuse about the logical and technical principles are even going back to 1934.

Konrad Zuse, also created the first programming language of the world, called the Plankalkül. (1942-1945)
F. L. Bauer (University of München)

Meetings with Zuse



Which tenets?

Zuse: “Those which can be understood by an Engineer“.

But many years passed before the deterministic approach of Gerard ‘t Hooft (2002) made a complete elaboration of the originally conceived ideas possible, namely that of

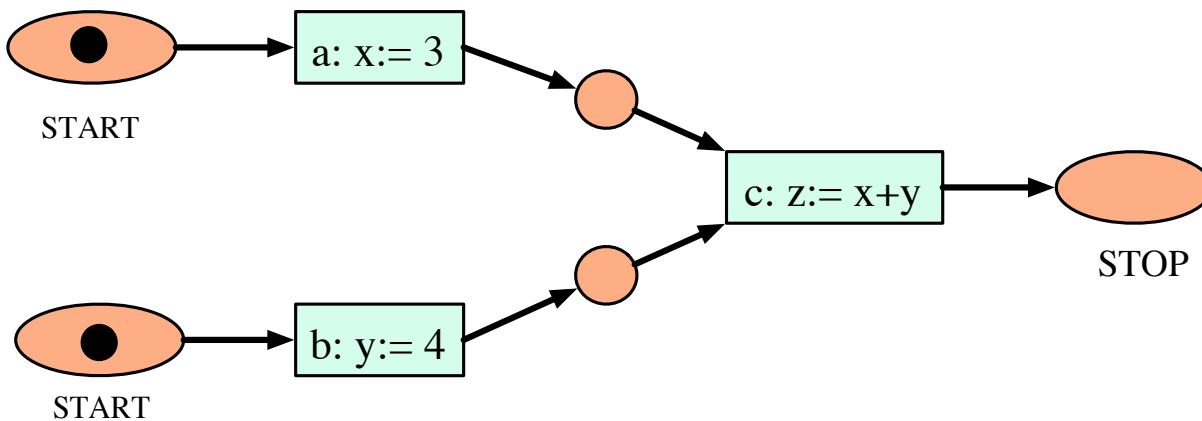
Combinatorial Modelling

weiter:

67

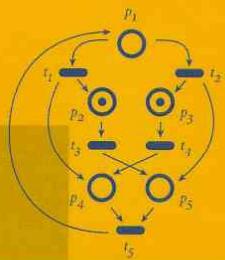
[http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/
profs/petri.html](http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/profs/petri.html)

Allgemeine Eigenschaften von Petrinetzen



- Rechnerwerkzeuge für Editieren, Simulation und Analyse
siehe z.B. TGI-Tool RENEW <http://www.renew.de/>
- Universalität in Anwendbarkeit (Anwendungen in fast allen Gebieten)
- Varianten des gleichen Modellierungskonzeptes (Zeit-Netze, stochastische Netze, high-level, objektorientiert, ...)

Claude Girault
Rüdiger Valk



Petri Nets for Systems Engineering

A Guide to Modeling, Verification,
and Applications



Springer

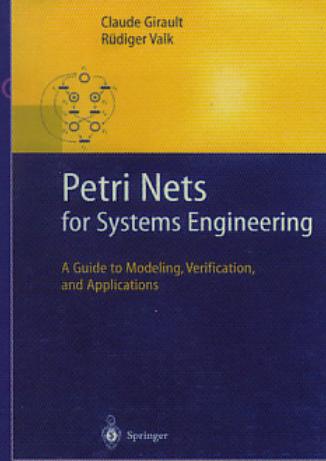
国外计算机科学教材系列

系统工程Petri网

— 建模、验证与应用指南

Petri Nets for Systems Engineering

A Guide to Modeling, Verification, and Applications



[法] Claude Girault 著
[德] Rüdiger Valk 译

王生原 余 鹏 霍金键 译
袁崇义 审校

33rd International Conference on
Application and Theory of Petri Nets and Concurrency

12th International Conference on
Application of Concurrency to System Design

Organising Committee
General and Organizing Chair
D. Moldt

Call for Papers and Announcement

Petri Nets 2013

34th INTERNATIONAL CONFERENCE ON APPLICATION AND THEORY OF PETRI NETS AND CONCURRENCY

Milano, Italy, June 24–28, 2013

Additional information about the conference will be published via
<http://www.mc3.disco.unimib.it/petrinets2013/>

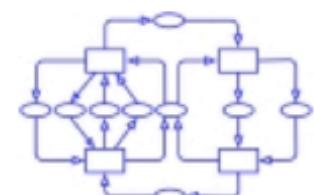
Contact e-mail: petrinets2013@disco.unimib.it

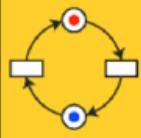
Important Dates:

Submission of Workshop Proposals:	June 7, 2012
Submission of Tutorial Proposals:	January 10, 2013
Submission of Papers:	January 10, 2013
Notification:	March 1, 2013
Final Version Due:	April 1, 2013
Participation in Tool Exhibition:	June 1, 2013
Workshops & Tutorials:	June 24–25, 2013
Conference:	June 26–28, 2013

The deadline for submission of papers is STRICT. However, if you submit the title page by January 10 it is sufficient to submit the full paper by January 15.

Some of the best papers accepted for the conference will be invited as submissions to a special issue of the [Fundamenta Informaticae](#) journal.

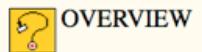




Welcome to the Petri Nets World

The purpose of the *Petri Nets World* is to provide a variety of online services for the international Petri Nets community. The services constitute, among other things, information on the *International Conferences on Application and Theory of Petri Nets*, mailing lists, bibliographies, tool databases, newsletters, and addresses. The [Petri Nets Steering Committee](#) supervises these activities, and the site is maintained by the [TGI group](#) at the University of Hamburg, Germany.

Contents of the Petri Nets World



OVERVIEW

- [Introductions](#) •

Papers, books, and tutorials which introduce the basic concepts of Petri Nets.

- [Frequently Asked Questions](#) •

Typical occurring questions about Petri Nets. Who is *Carl Adam Petri*? Which kinds of Petri Nets exist? How is "Petri Nets" written in different languages?

- [Related Links](#) •

Contains miscellaneous links to external Web pages about Petri Nets, related topics, *search engines*, and special interest groups.



APPLICATIONS and ARTIFACTS

- [Tools and Software](#) •

Search databases for software packages which support Petri Nets. Also find several online Petri Nets *Java Applets*.

- [Success Stories](#) •

Case studies on Petri Nets applications in academia and industry.

- [Standardisation](#) •

Follow the current progress of the work on making an ISO standard for high-level Petri Nets. Note also the activity on designing an *interchange format* for Petri Nets.



PUBLICATIONS

- [Bibliographies](#) •

Search large publication databases on Petri Nets. In particular the *Petri Nets Bibliography* contains thousands of entries.

- [Periodicals](#) •

Information on the *Petri Net Newsletter*. Also contains a listing of similar journals, newsletters, and other regularly published material.



ACADEMIA

- [Research Activities](#) •

Contains a list of Petri Nets research groups and projects from academia and industry.

- [Education with Petri Nets](#) •

Resources for teachers and students from academia and industry. Also contains information on *advanced courses on Petri Nets*, and other courses and schools.



FORUMS

- [Meetings and Events](#) •

Calendar of symposia, conferences, workshops, and similar events. Contains information on important events such as the *Conferences on Application and Theory of Petri Nets* or the *Workshops on Petri Nets and Performance Models*. Events announced on the PetriNets Mailing List are also listed here.

- [Mailing Lists](#) •

Primary forums for announcements and discussions on all topics on Petri Nets. View subscription information and

Selected News in the Petri Nets World

[November 11, 2012:](#) Call for Papers for the International Conference on Business Process Modeling, Development, and Support (BPMDS'2013).

[November 04, 2012:](#) Second Call for Model for the Model Checking Contest 2013.

[October 26, 2012:](#) PhD position available in model-driven software engineering and verification at Bergen University College, Faculty of Engineering.

[October 26, 2012:](#) First Call for Papers for Logics, Agents and Mobility (LAM'13) in Exeter, United Kingdom, April 2-5, 2013.

[October 24, 2012:](#) First Call for Papers for TAP 2013 in Budapest, Hungary, June 17-21, 2013.

[October 12, 2012:](#) ePNK version 1.0.0 released.

[October 11, 2012:](#) WoPeD 3.0.0 has been released.

[October 10, 2012:](#) The first BPM Newsletter can now be downloaded.

[October 01, 2012:](#) Call for Book Chapters - Theory and Application of Multi-Formalism Modeling.

[October 01, 2012:](#) First Call for Papers for Integrated Formal Methods 2013 in Turku, Finland.

[September 26, 2012:](#) Call for Papers for Petri Nets 2013 in Milano, Italy.

[September 25, 2012:](#) Call for Papers ACSD 2013 in Barcelona, Spain.

[September 24, 2012:](#) PhD position in model-driven software engineering and verification available in Bergen University College.

[September 20, 2012:](#) Invitation to organise Petri Net conferences in 2015 and 2016.

[September 20, 2012:](#) Call for Papers for the 25th International Conference on Advanced Information Systems Engineering (CAiSE 2013) in Valencia, Spain.

[September 18, 2012:](#) Second Call for Papers for the 8th Australasian Ontology Workshop (AOW 2012).

[September 17, 2012:](#) Final Call for Participation for the 7th IFIP Conference on Theoretical Computer Science 2012.

[September 15, 2012:](#) Call for Nominations for the SPEC Distinguished Dissertation Award 2012.

[September 14, 2012:](#) 2 Ph.D. Positions in Computer Security available at the University of Luxembourg.

[September 12, 2012:](#) The raw report of the model checking contest from Petri Nets 2012 is now online.

[September 10, 2012:](#) PhD positions in Computer Science available at Univ. Padova, Italy. The deadline is Oct. 8, 2012.

[September 07, 2012:](#) Call for Papers for the special session on Process Mining at CIDM 2013, Singapore.

[September 05, 2012:](#) PhD positions in Formal Software Modeling and Verification available at RWTH Aachen University.

[September 04, 2012:](#) Last Call for Participation for ESORICS 2012 in Pisa, Italy.

[September 03, 2012:](#) Call for Participation for Reachability Problems 2012, September 17-19, 2012, in Bordeaux, France.

[September 03, 2012:](#) First Call for Papers for the 24th International Conference on Automated Deduction June 9-14, 2013, Lake Placid, New York, USA.

FGI 2

Daniel Moldt

Petrinetze: Entwicklung

Drei Ansätze zur Einführung von Petrinetzen:

- ➊ Entwicklung aus Programm-Code
- ➋ Kommunizierende Automaten
- ➌ von einfachen zu höheren Netzen

Ausgangspunkt: Programm

starting from program code

→ start

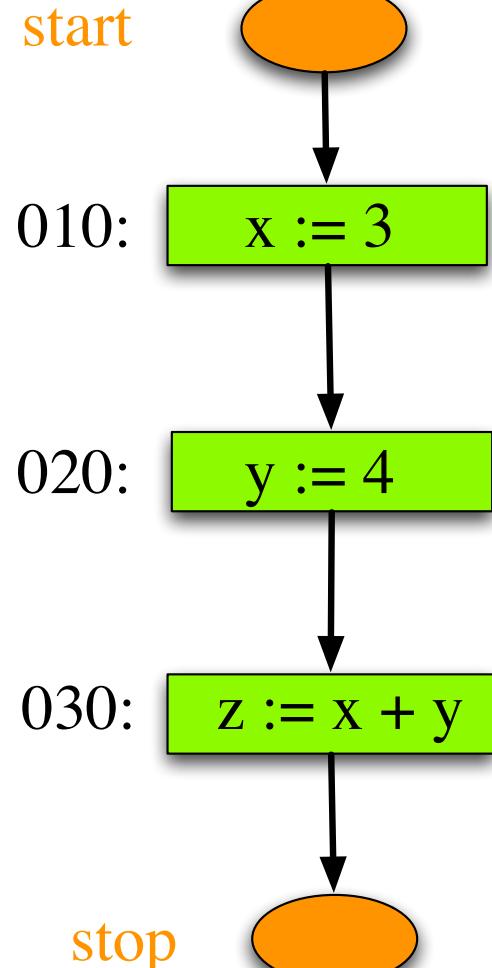
010: $x := 3$

020: $y := 4$

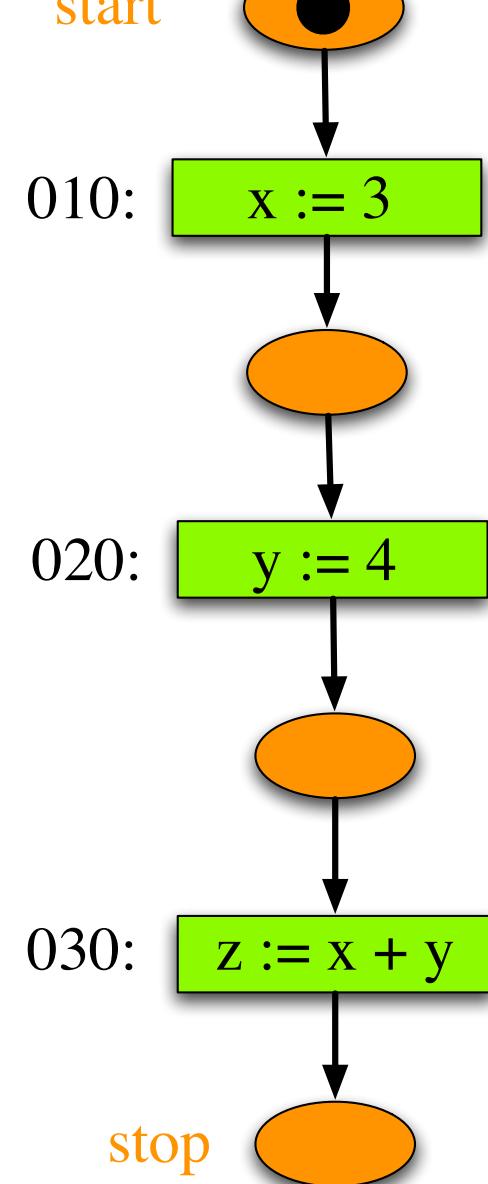
030: $z := x + y$

stop

start



start



Ausgangspunkt: Programm

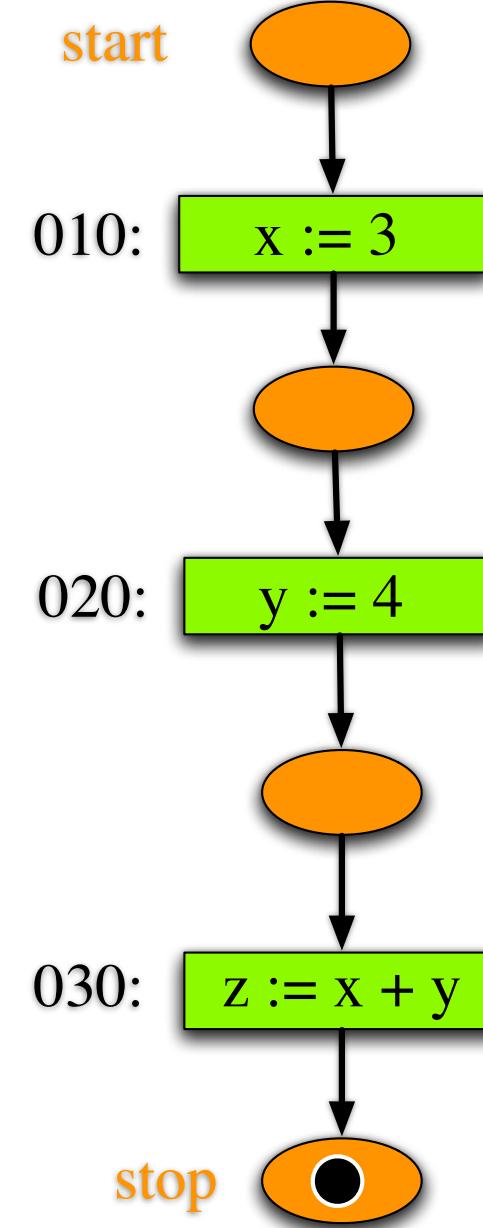
start

010: $x := 3$

020: $y := 4$

030: $z := x + y$

stop



Ausgangspunkt: Programm

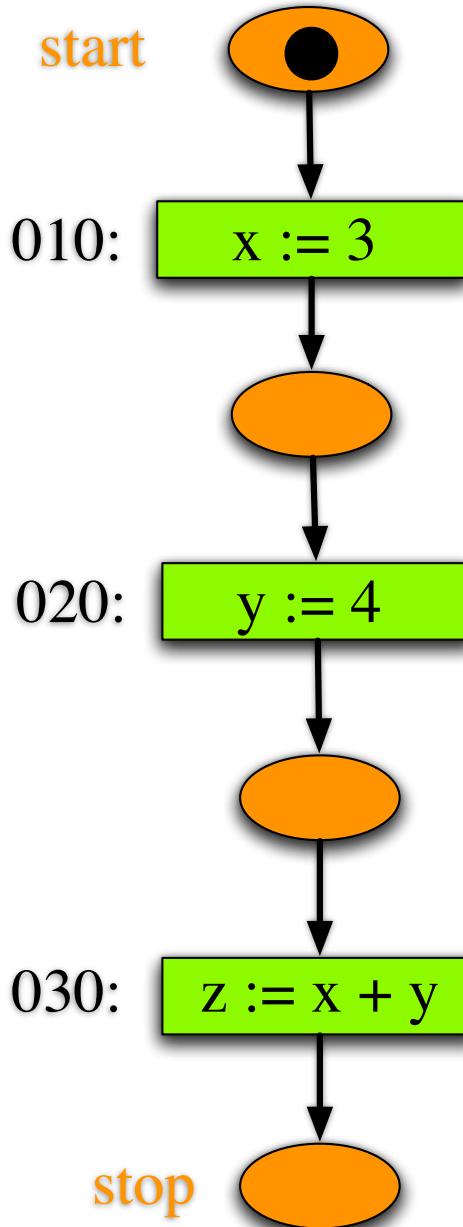
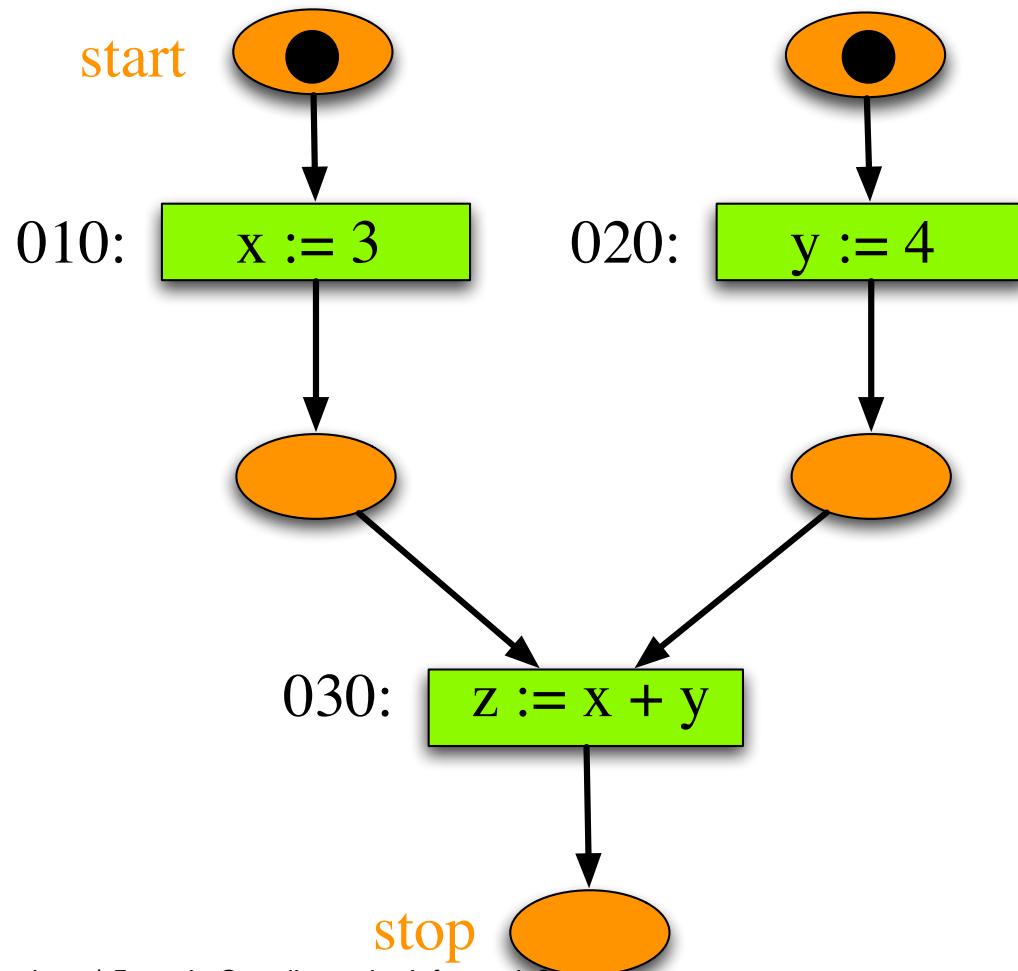
start

010: $x := 3$

020: $y := 4$

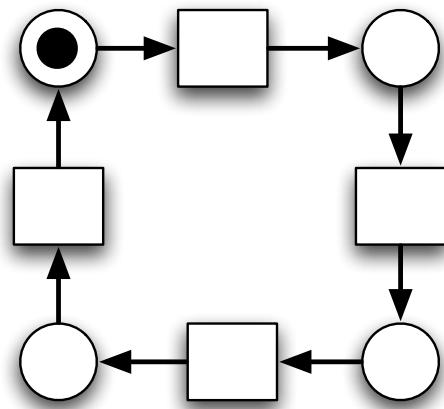
030: $z := x + y$

stop

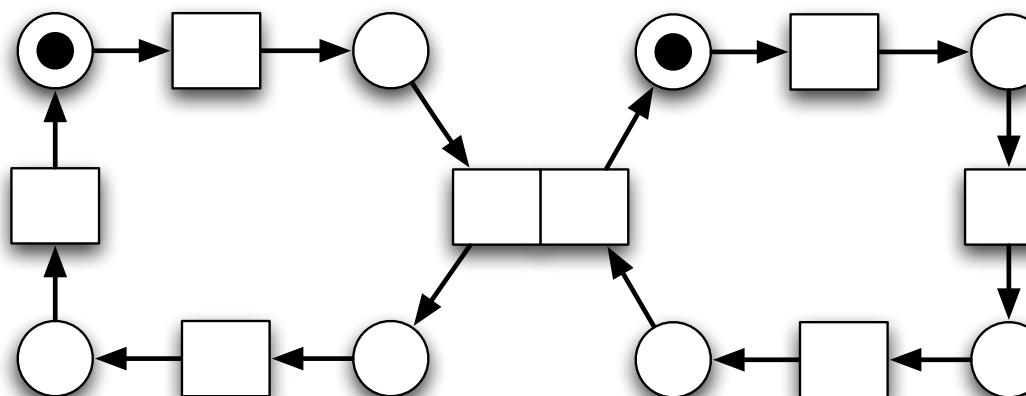
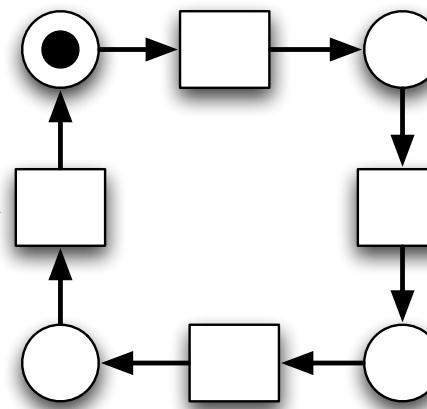


Ausgangspunkt: Kommunizierende Automaten

Automat 1



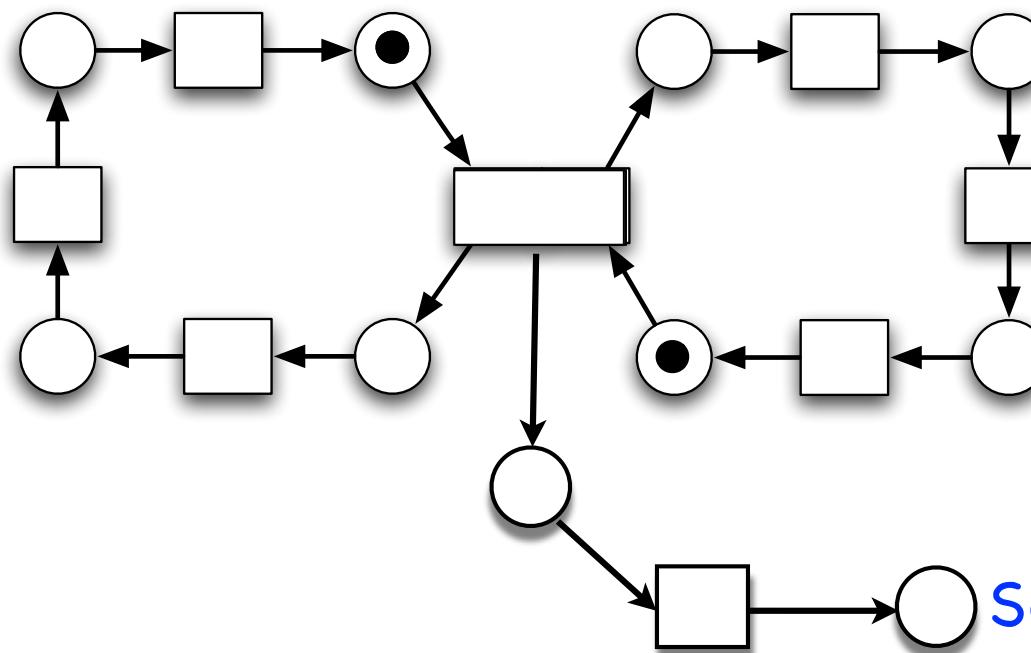
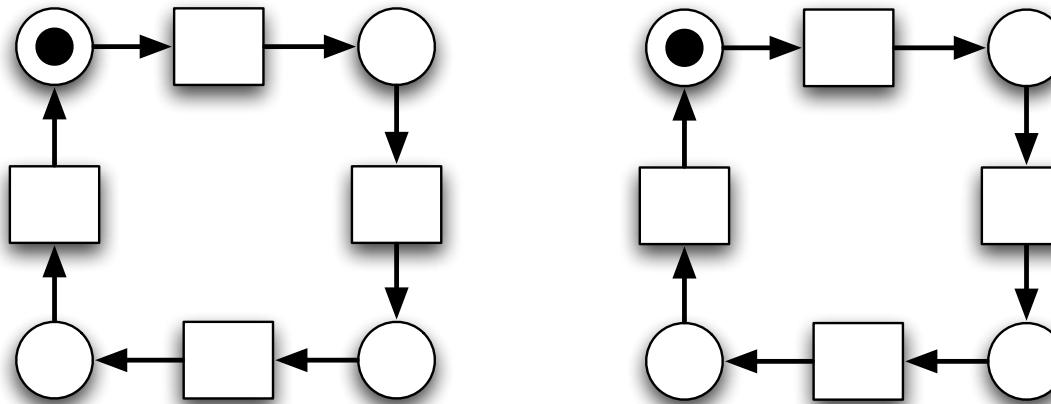
Automat 2



kommunizierende Automaten

Schalten des Netzes

Ausgangspunkt: Kommunizierende Automaten



FGI 2

Daniel Moldt

Grundprinzipien

Grundprinzipien der Petrinetze

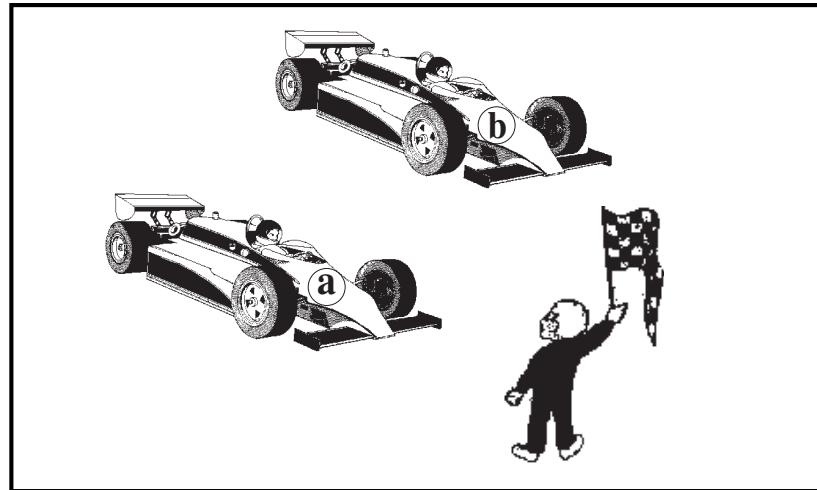
Einführung der wichtigsten Prinzipien
an einem einfachen Beispiel:

- Lokalität
- Nebenläufigkeit
- Graphische und formaltextuelle Darstellung
- von einfachen zu höheren Petrinetzen

Das Rennwagen-Beispiel

Bedingungen

- p_1 : car a; preparing for start
- p_2 : car a; waiting for start
- p_3 : car a; running
- p_4 : ready sign of car a
- p_5 : start sign for car a
- p_6 : starter; waiting for ready signs
- p_7 : starter; start sign given
- p_8 : ready sign of car b
- p_9 : start sign for car b
- p_{10} : car b; preparing for start
- p_{11} : car b; waiting for start
- p_{12} : car b; running

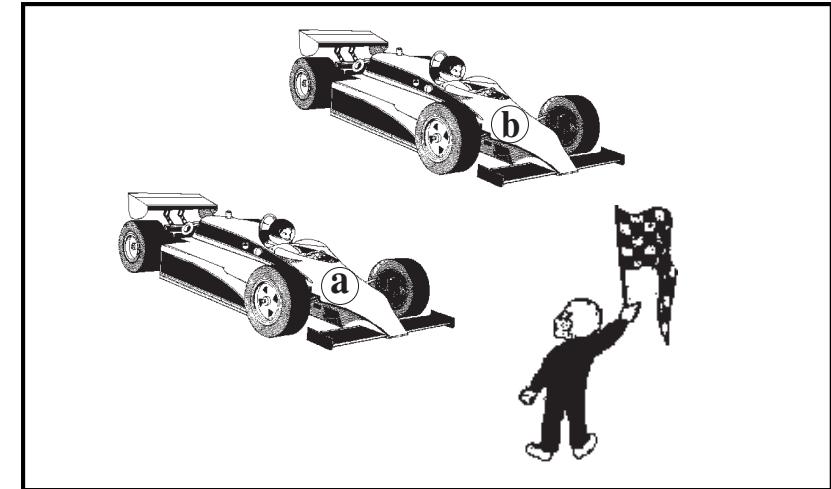


Aktionen

- t_1 : car a; send ready sign
- t_2 : car a; start race
- t_3 : starter; give start sign
- t_4 : car b; send ready sign
- t_5 : car b; start race

Das Rennwagen-Beispiel

globaler Zustandsübergang Transitionssystem



car a; preparing for start

car a; waiting for start

car a; running

ready sign of car a

start sign for car a

starter; waiting for ready s.

starter; start sign given

ready sign of car b

start sign for car b

car b; preparing for start

car b; waiting for start

car b; running

p1=true

p2

p3

p4

p5

p6=true

p7

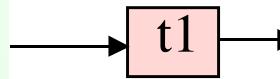
p8

p9

p10=true

p11

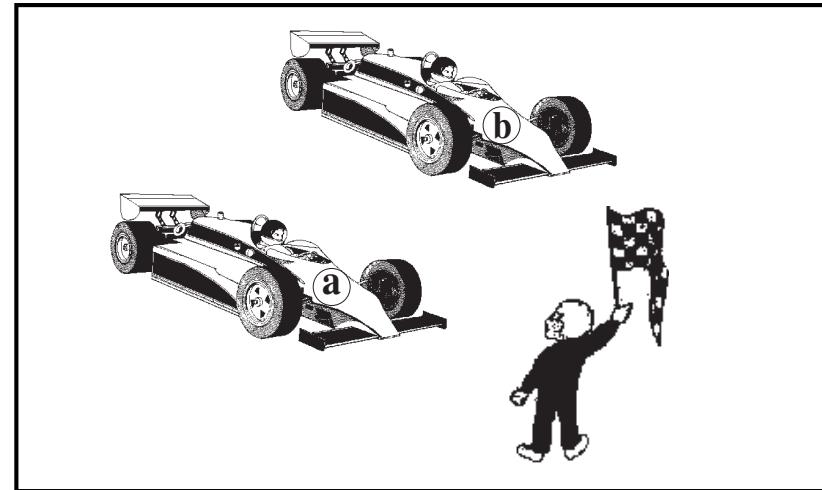
p12



p1 car a; preparing for start
p2=true car a; waiting for start
p3 car a; running
p4=true ready sign of car a
p5 start sign for car a
p6=true starter; waiting for ready s.
p7 starter; start sign given
p8 ready sign of car b
p9 start sign for car b
p10=true car b; preparing for start
p11 car b; waiting for start
p12 car b; running

Das Rennwagen-Beispiel

lokaler Zustandsübergang Transitionssystem



car a; preparing for start
car a; waiting for start
car a; running
ready sign of car a
start sign for car a
starter; waiting for ready s.
starter; start sign given
ready sign of car b
start sign for car b
car b; preparing for start
car b; waiting for start
car b; running

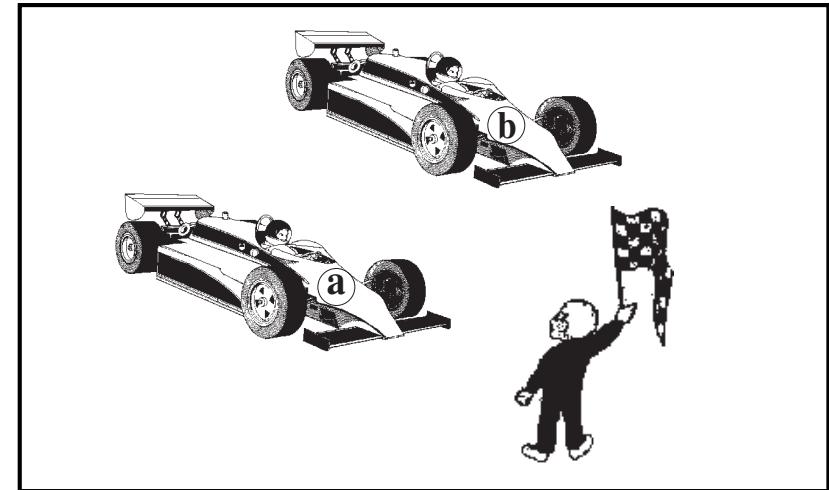
p1=true
p2
p3
p4
p5
p6=true
p7
p8
p9
p10=true
p11
p12

p1
p2=true
p3
p4=true
p5
p6=true
p7
p8
p9
p10=true
p11
p12

car a; preparing for start
car a; waiting for start
car a; running
ready sign of car a
start sign for car a
starter; waiting for ready s.
starter; start sign given
ready sign of car b
start sign for car b
car b; preparing for start
car b; waiting for start
car b; running

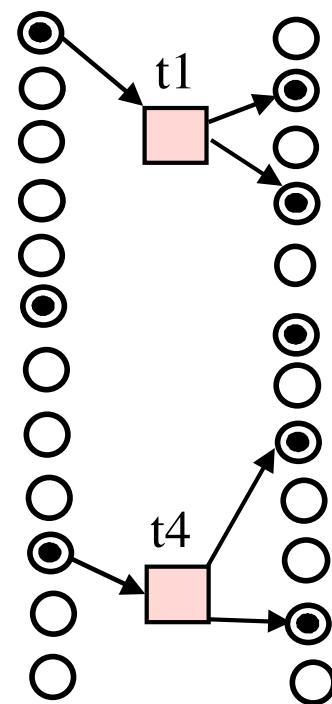
Das Rennwagen-Beispiel

unabhängiger Zustandsübergang
nebenläufiger Zustandsübergang



car a; preparing for start
car a; waiting for start
car a; running
ready sign of car a
start sign for car a
starter; waiting for ready s.
starter; start sign given
ready sign of car b
start sign for car b
car b; preparing for start
car b; waiting for start
car b; running

p1=true
p2
p3
p4
p5
p6=true
p7
p8
p9
p10=true
p11
p12



p1
p2=true
p3
p4=true
p5
p6=true
p7
p8=true
p9
p10
p11=true
p12

car a; preparing for start
car a; waiting for start
car a; running
ready sign of car a
start sign for car a
starter; waiting for ready s.
starter; start sign given
ready sign of car b
start sign for car b
car b; preparing for start
car b; waiting for start
car b; running

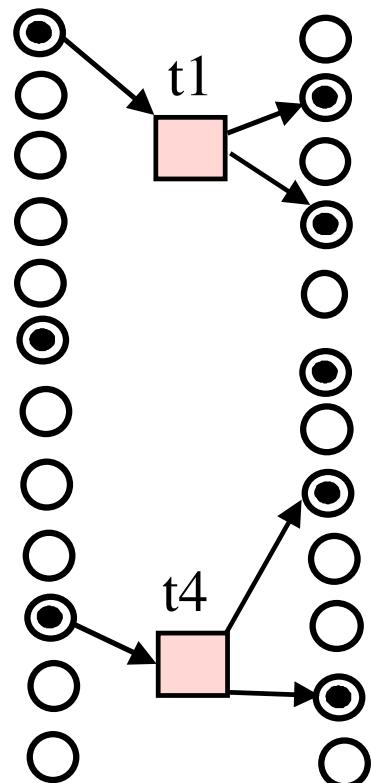
I

Das Prinzip der Dualität in Netzen

Es gibt zwei disjunkte Mengen von Grundelementen:
P-Elemente (state elements, Plätze, Stellen) und
T-Elemente (Transition-Element, Transistionen).

Dinge der realen Welt werden entweder als **passive Elemente** aufgefasst und als P-Elemente dargestellt (z.B. Bedingungen, Plätze, Betriebsmittel, Wartepools, Kanäle usw.) oder als **aktive Elemente**, die durch T-Elemente repräsentiert werden (z.B. Ereignisse, Aktionen, Ausführungen von Anweisungen, Übermitteln von Nachrichten usw.).

Lokalität und Nebenläufigkeit



II

Das Prinzip der Lokalität für Petri Netze

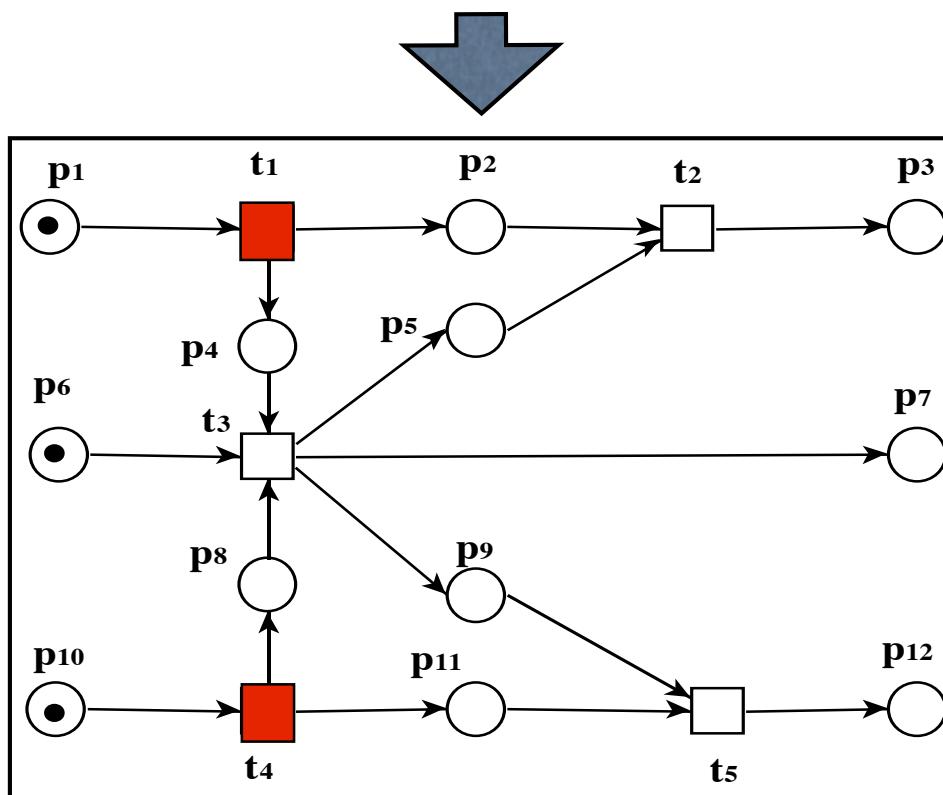
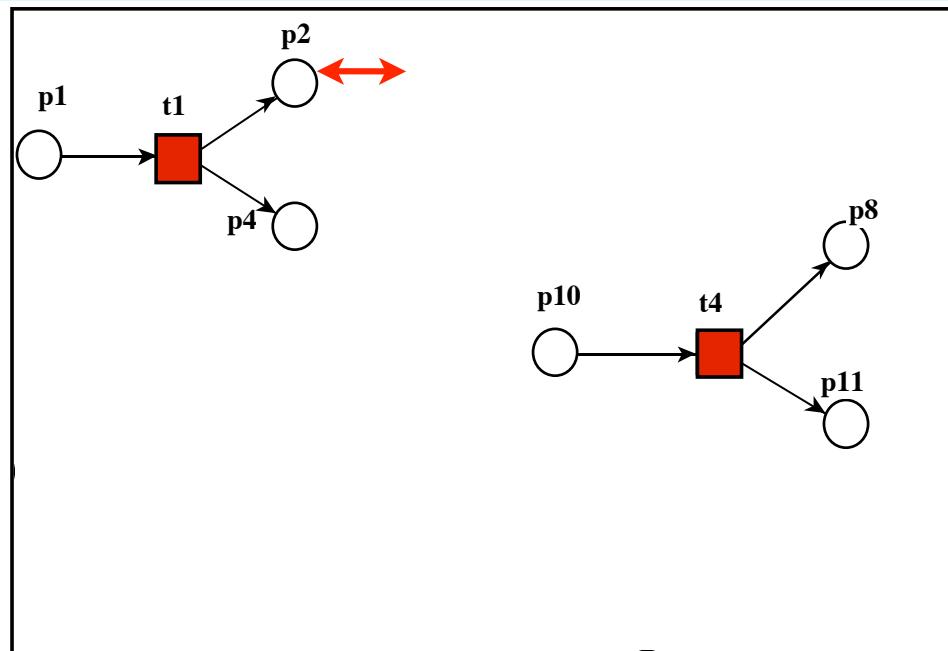
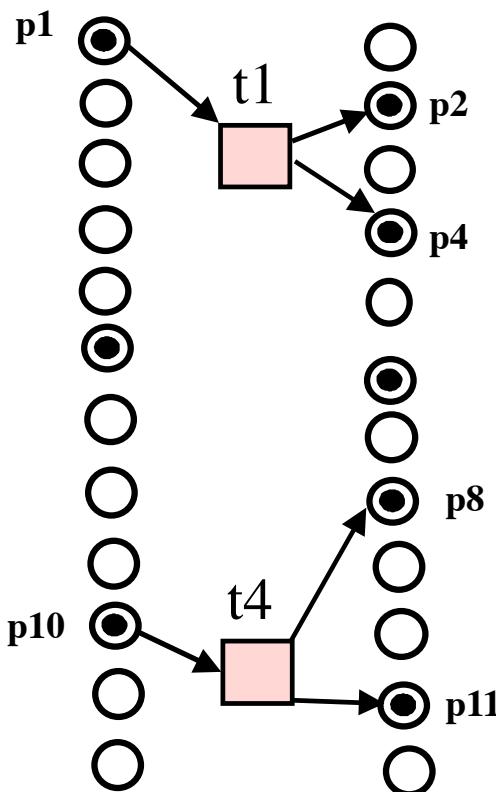
Das Verhalten einer Transition wird ausschließlich durch ihre **Lokalität** bestimmt, welche sich aus ihr und der Gesamtheit ihrer Eingangs- und Ausgangs-Elemente zusammensetzt.

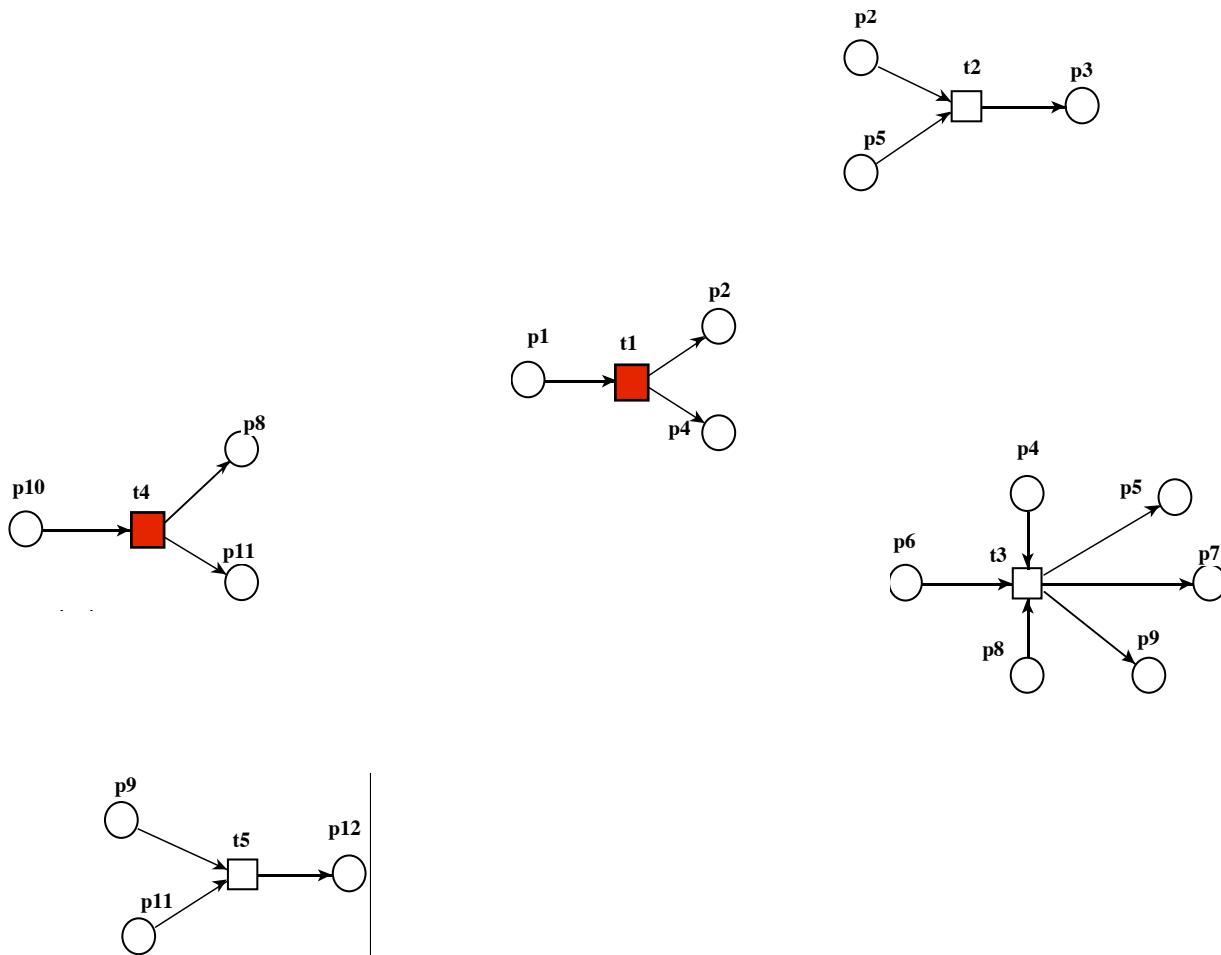
III

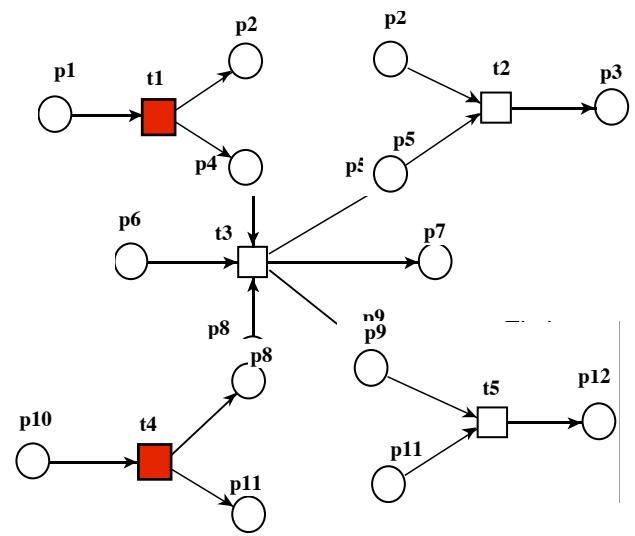
Das Prinzip der Nebenläufigkeit (concurrency) für Petrinetze

Transitionen mit disjunkter Lokalität finden unabhängig (**nebenläufig**, concurrently) statt.

Netzkomposition







Darstellung von Petrinetzen

IV

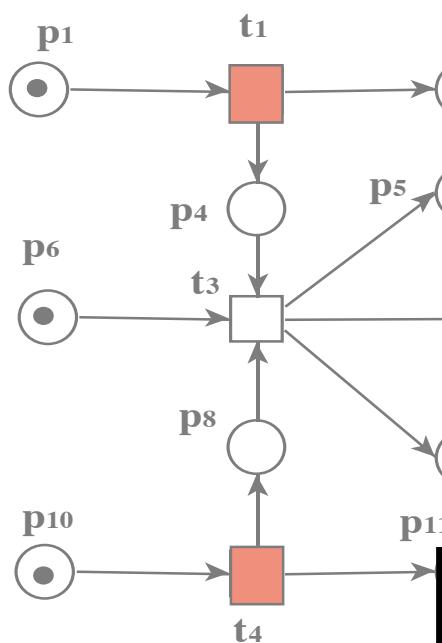
Das Prinzip der grafischen Darstellung von Petrinetzen

P-Elemente (auch „S-Elemente“) werden durch **runde** grafische Elemente (Kreise, Ellipsen,...) dargestellt (rund wie im Buchstaben P oder S).

T-Elemente werden durch **eckige** grafische Elemente (Rechtecke, Balken,...) dargestellt (eckig wie im Buchstaben T).

Kanten (auch „Pfeile“) verbinden T-Elemente mit den P-Elementen ihrer Lokalität.

Also gibt es nur Kanten zwischen T-Elementen und P-Elementen.
Außerdem gibt es Beschriftungen, wie Bezeichner, Gewichtungen oder Schutzbedingungen (guards).



V

Das Prinzip der formaltextuellen Darstellung von Petrinetzen

Zu jeder graphischen Darstellung eines Petrinetzes gibt es eine **äquivalente** formaltextuelle Darstellung und umgekehrt.

Petrinetz

Definition: Ein *Netz* ist ein Tripel $\mathcal{N} = (P, T, F)$, wobei gilt:

- P ist eine Menge von *Plätzen* (auch: *Stellen*).
- T eine Menge von *Transitionen* (disjunkt zu P).
- $F \subseteq (P \times T) \cup (T \times P)$ ist die *Flussrelation*.

Falls P und T endlich sind, dann heißt auch das Netz \mathcal{N} endlich.

$$(p_6, t_3) \in F \quad (t_3, p_5) \in F$$

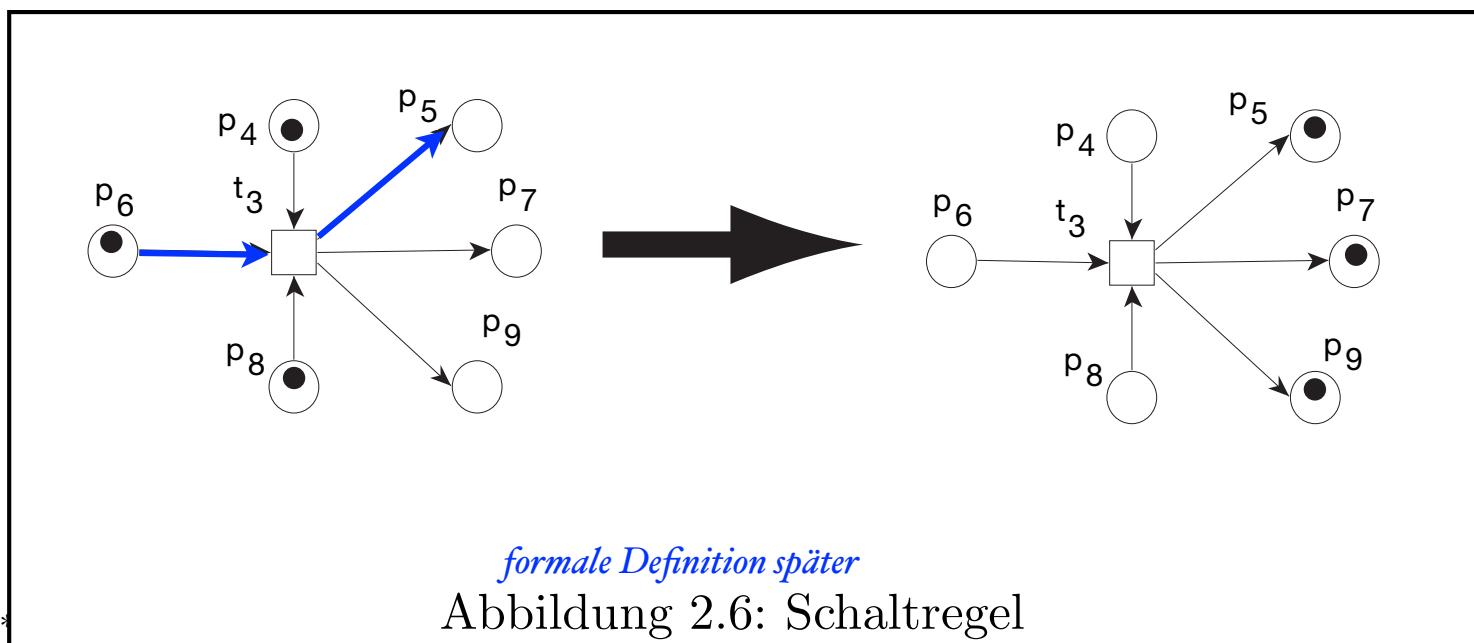


Abbildung 2.6: Schaltregel

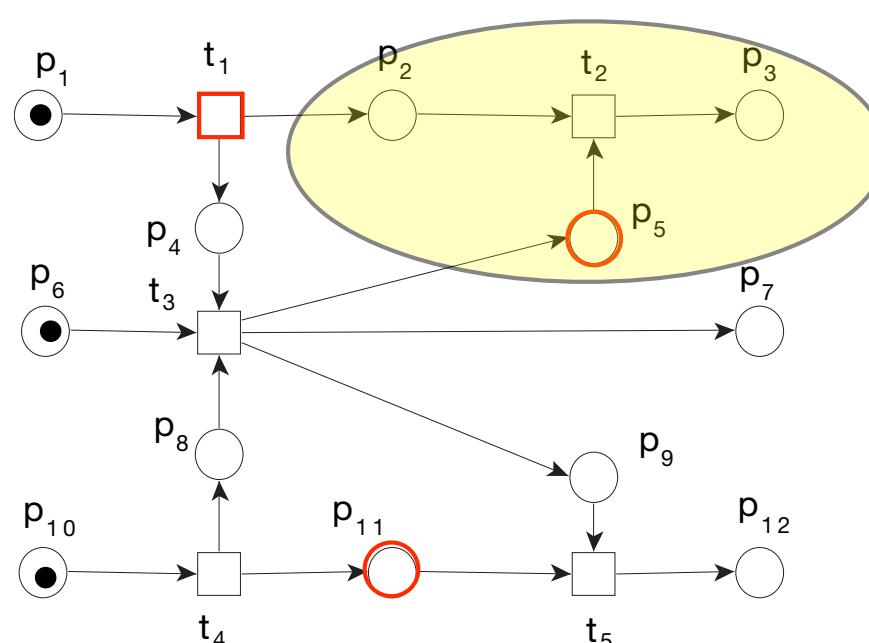
Eingang, Ausgang, Lokalität

Definition: Für ein Netz $\mathcal{N} = (P, T, F)$ und ein Element $x \in P \cup T$ bezeichnet $\bullet x := \{y \in P \cup T \mid (y, x) \in F\}$ die Menge der *Eingangselemente* und $x^\bullet := \{y \in P \cup T \mid (x, y) \in F\}$ die Menge der *Ausgangsselemente* von x .

Für eine Menge von Elementen $X \subseteq P \cup T$ seien entsprechend

$$\bullet X := \bigcup_{x \in X} \bullet x \text{ und } X^\bullet := \bigcup_{x \in X} x^\bullet$$

$loc(y) := \{y\} \cup \bullet y \cup y^\bullet$ heißt *Lokalität* des Platzes oder der Transition y .



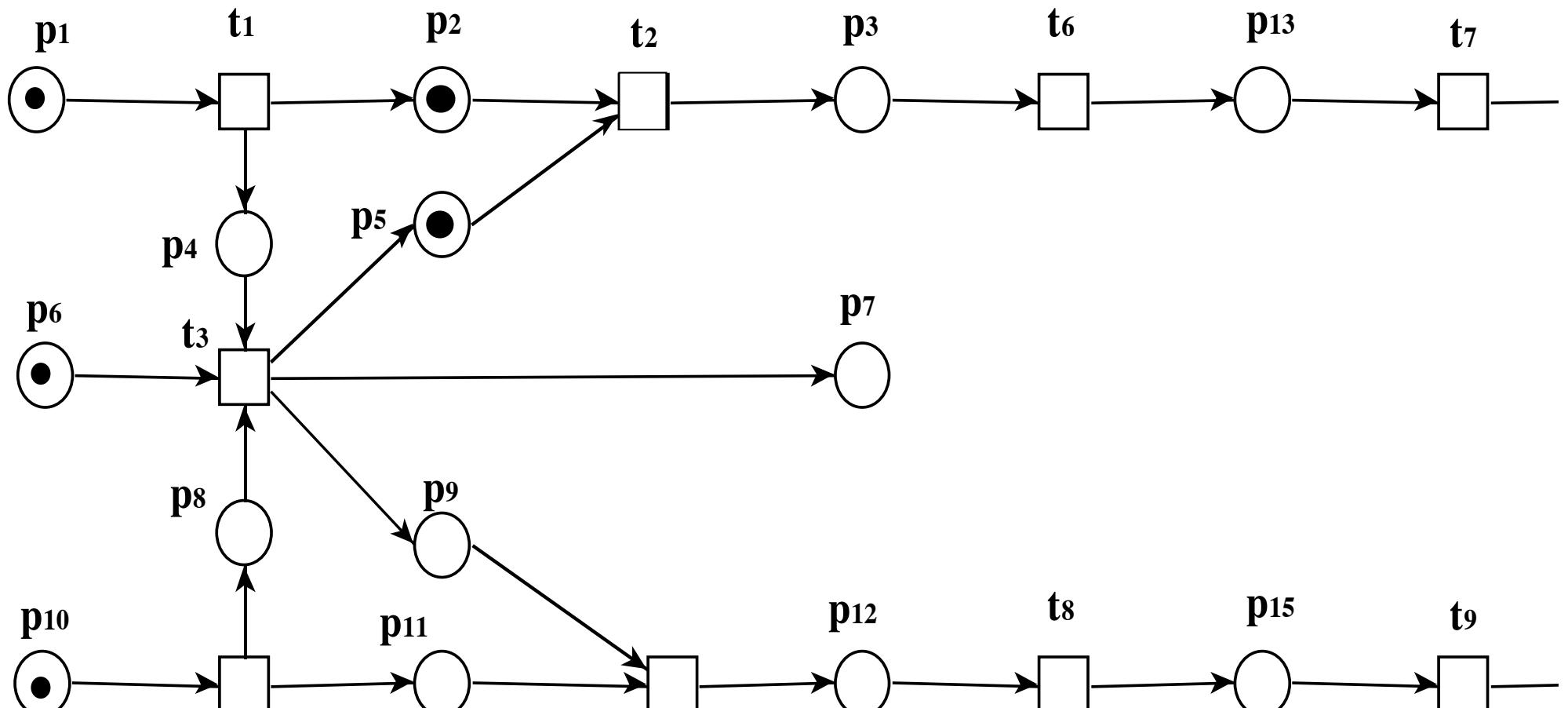
Beispiel: Für $A = \{t_1, p_5, p_{11}\}$ im Netz 3.5 erhält man $\bullet A = \{p_1, t_3, t_4\}$ and $A^\bullet = \{p_2, p_4, t_2, t_5\}$.

Markierung, Aktivierung, Schalten

Definition: Sei \mathcal{N} ein Petrinetz und $M \subseteq P$ eine Markierung.

Die Transition $t \in T$ ist in M aktiviert, gdw. $\bullet t \subseteq M$ und $t^\bullet \cap M = \emptyset$ gilt.

Die Nachfolgemarkierung M' ergibt sich dann als $M' = (M \setminus \bullet t) \cup t^\bullet$.

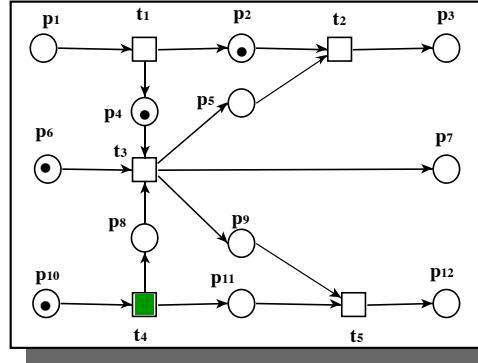
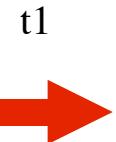
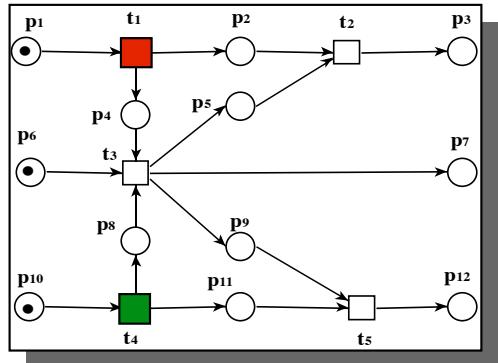


Ablaufsemantik

Wagen 1

Starter

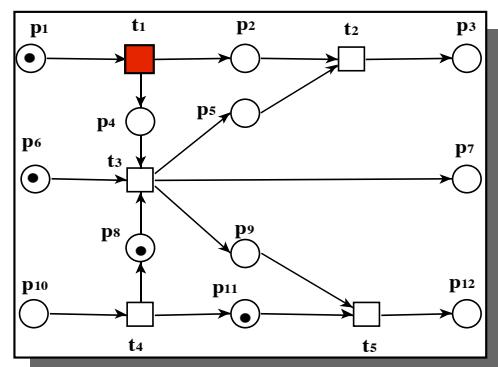
Wagen 2



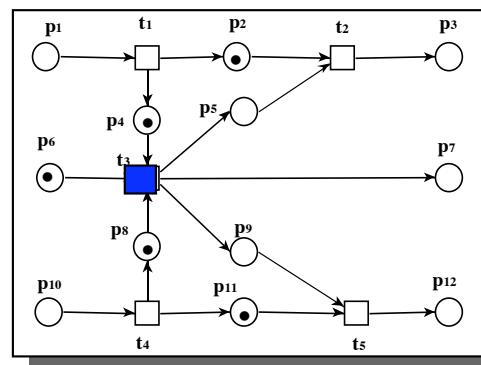
Ablaufsemantik:

- Folgen
- Schritte
- Prozess

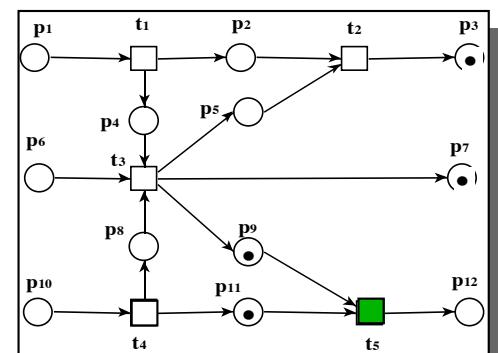
t4



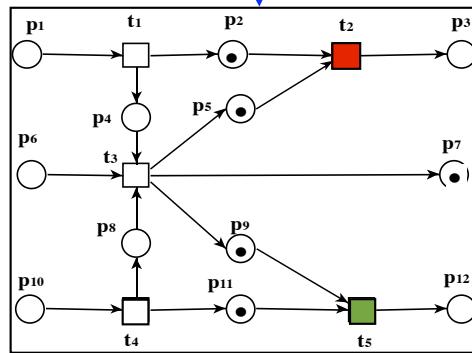
t1



t3



t2



Simultanes Schalten

Um das **simultane Schalten** zu beschreiben, wollen wir definieren, wann eine **Transitionsmenge** $U \subseteq T$ aktiviert ist.

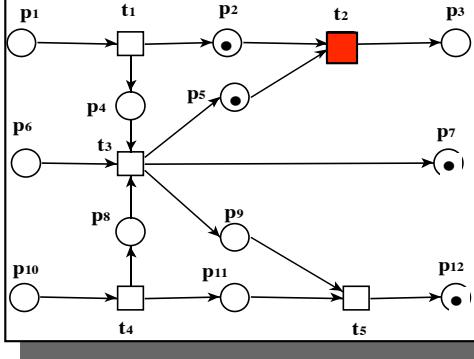
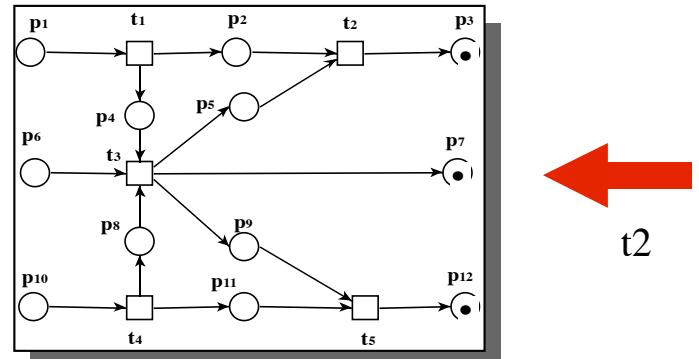
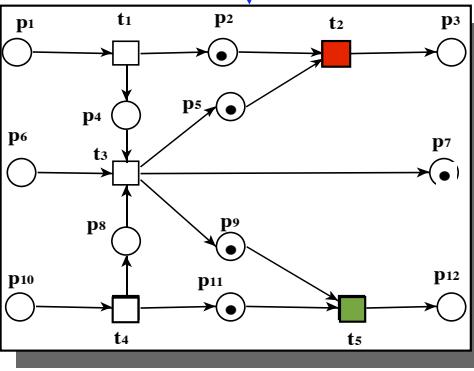
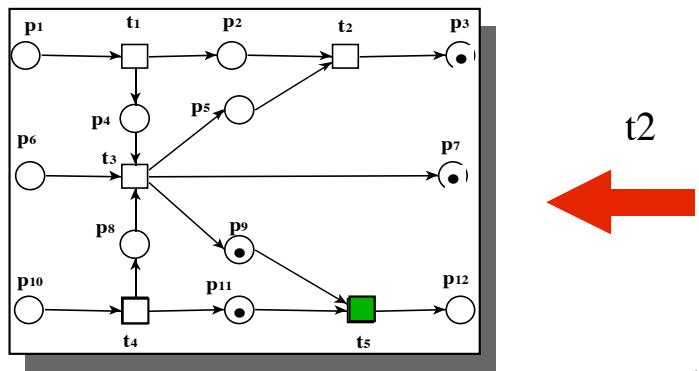
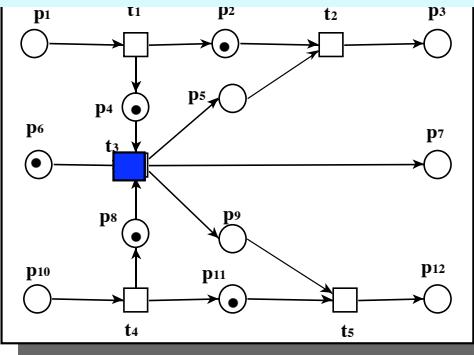
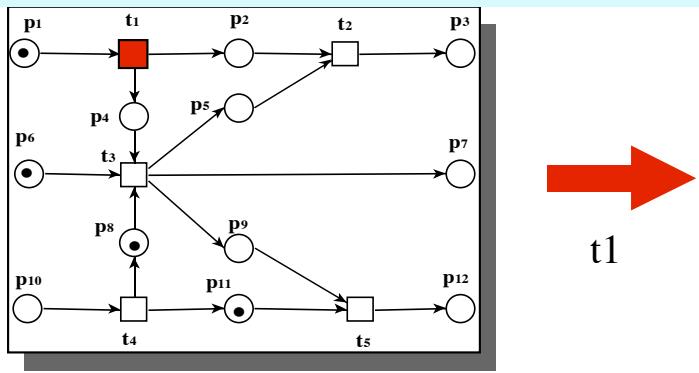
Definition: Sei \mathcal{N} ein Petrinetz und $M \subseteq P$ eine Markierung.

Eine Transitionsmenge $U \subseteq T$ ist in M aktiviert, wenn gilt:

1. Vorbereich markiert: $\bullet U \subseteq M$.
2. Nachbereich leer: $U^\bullet \cap M = \emptyset$.
3. Transitionen nebenläufig: $\forall t, t' \in U : t \neq t' \Rightarrow loc(t) \cap loc(t') = \emptyset$.

Die Nachfolgemarkierung M' ist definiert als $M' = (M \setminus \bullet U) \cup U^\bullet$.

Setzen wir $U = \{t\}$, so erhalten wir die bekannte sequentielle Schaltregel.

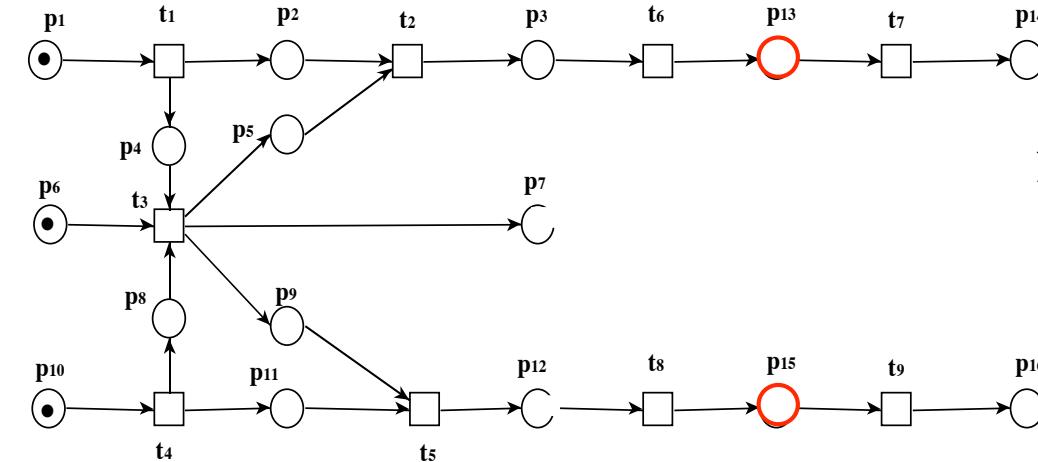


FGI 2

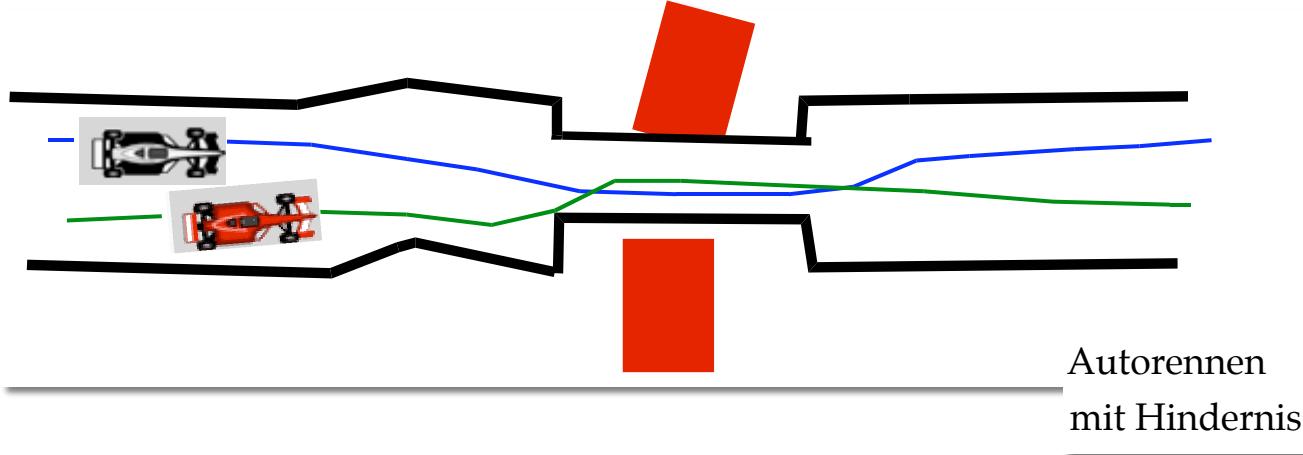
Daniel Moldt

Verfeinerung/Vergrößerung

Synchronisation



Erweiterung



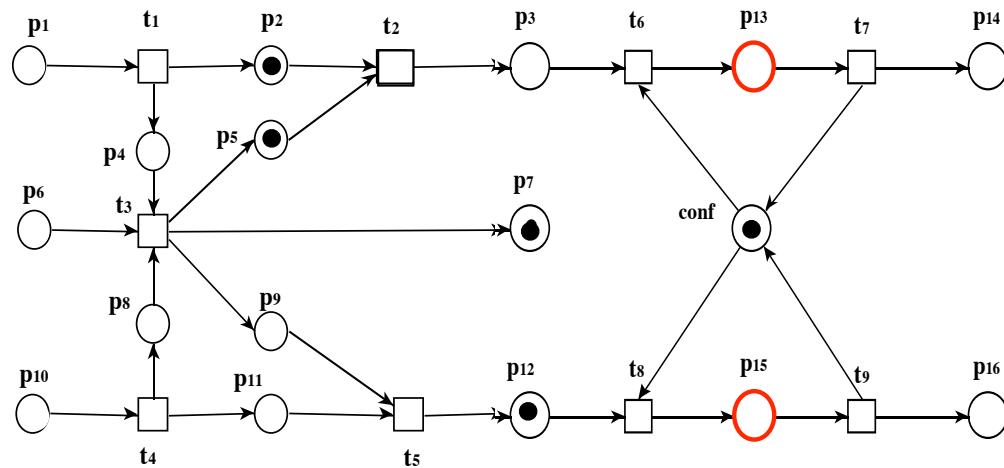
Autorennen
mit Hindernis

Java synchronized statement

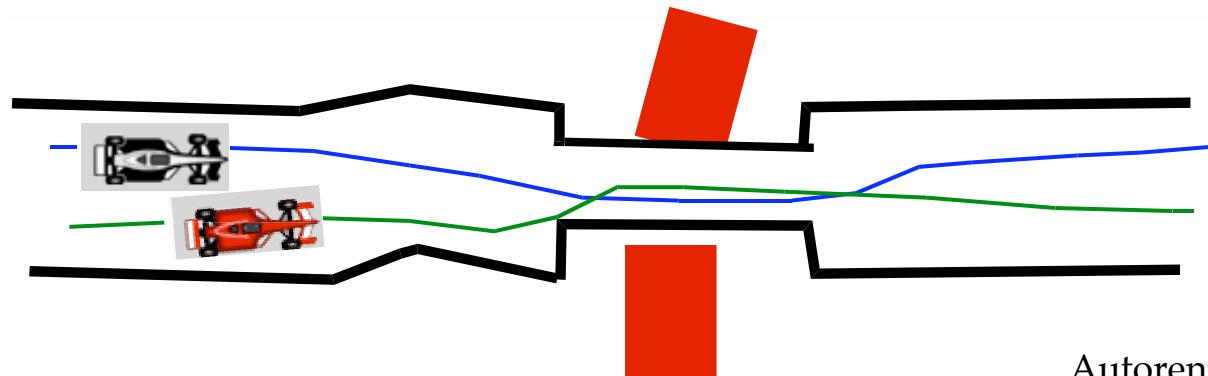
Access to an object may also be made mutually exclusive by using the **synchronized** statement:

```
synchronized (object) { statements }
```

Verfeinerung

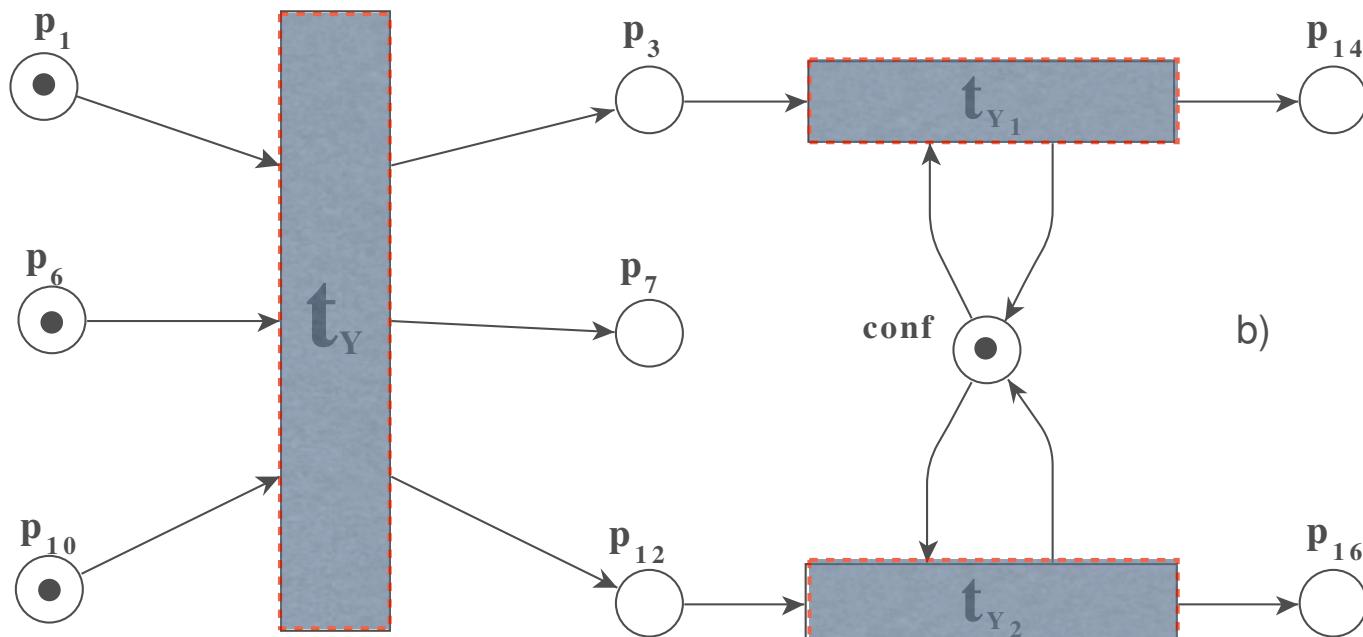
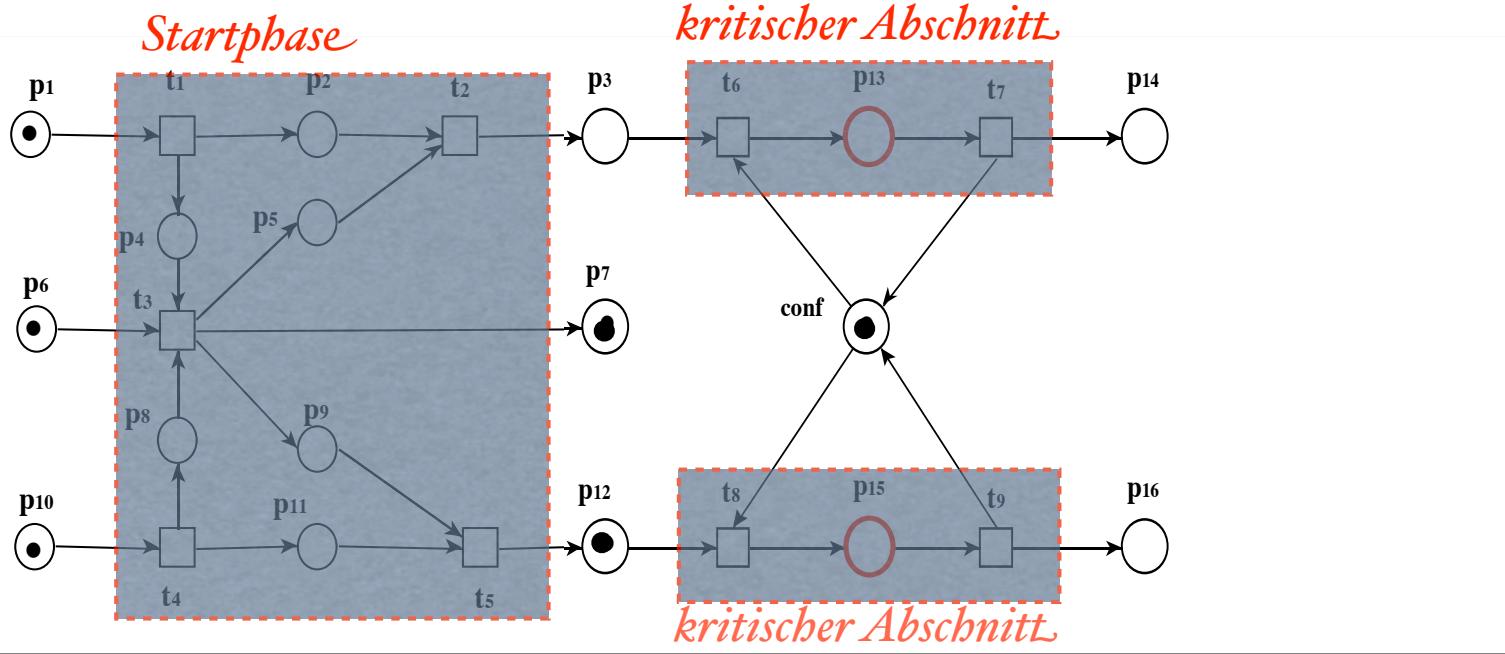


struktureller Konfliktplatz
↓
Verhaltenskonflikt
↓
wechselseitiger Ausschluss

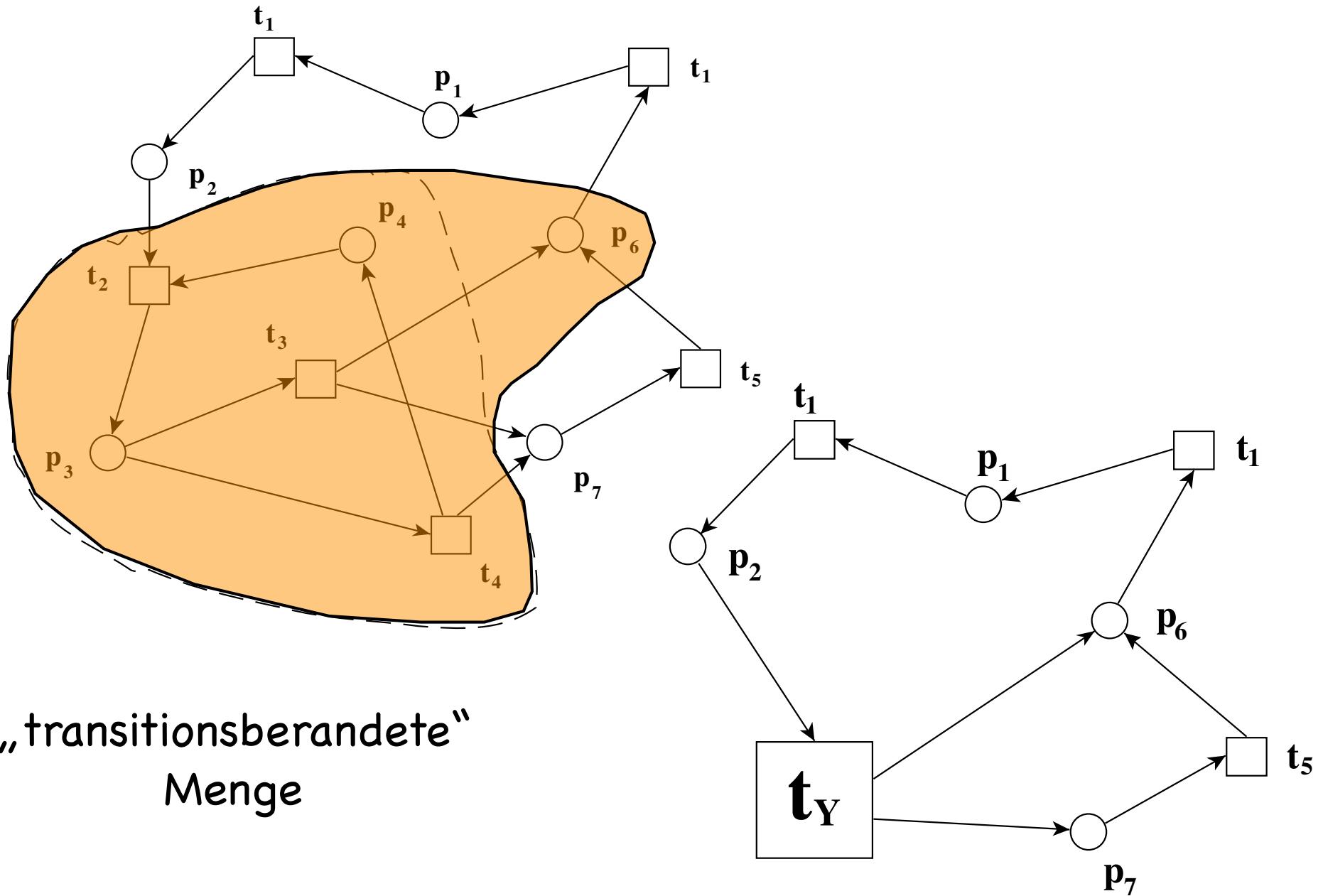


Autorennen
mit Hindernis

Vergrößerte Sicht



Beispiel



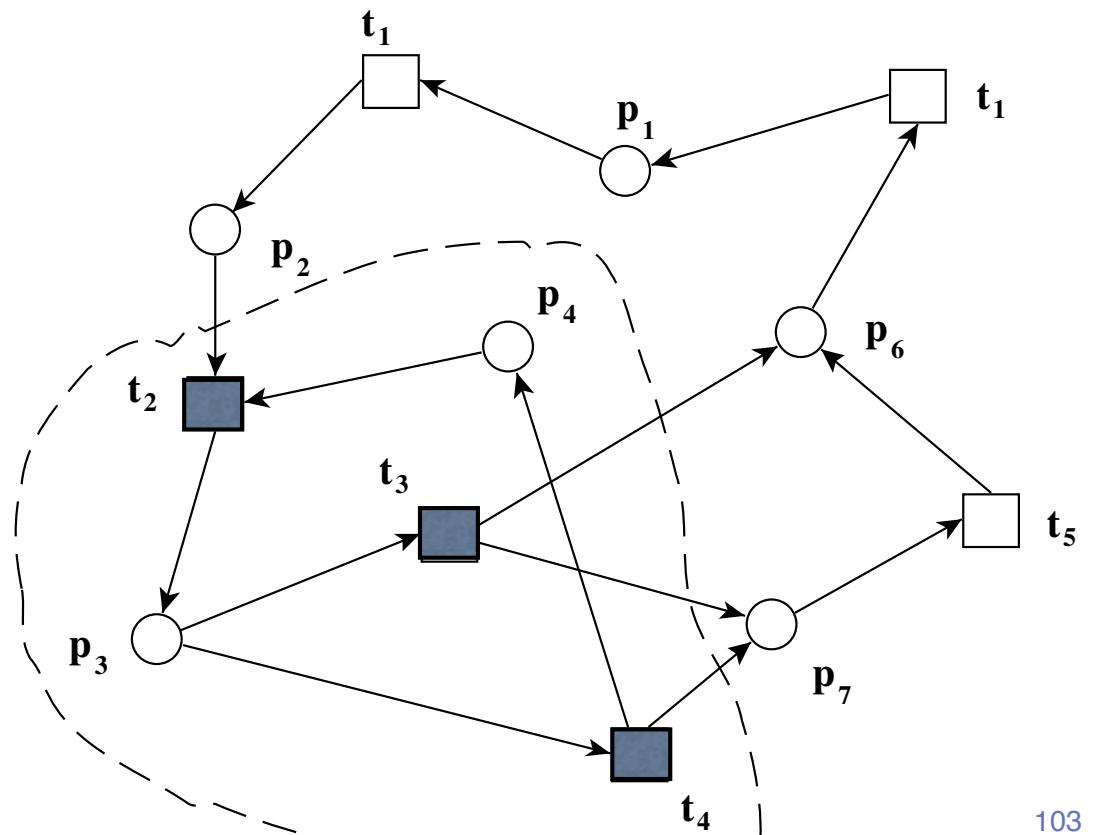
Berandete Mengen

Definition: Sei $\mathcal{N} = (P, T, F)$ ein Netz, $X := P \cup T$ und $Y \subseteq X$ eine Menge von Elementen. Dann heißt $\partial(Y) := \{y \in Y \mid \exists x \notin Y . x \in \text{loc}(y)\}$ der *Rand* (border) (der Menge Y).

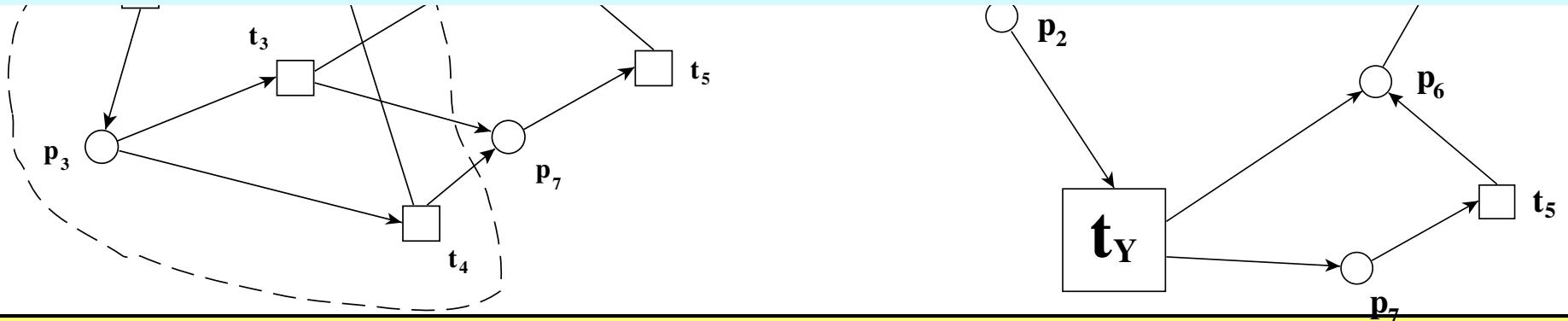
Y heißt *Platz-berandet* (place-bordered) oder *offen*, wenn $\partial(Y) \subseteq P$, und *Transitions-berandet* (transition-bordered) oder *abgeschlossen*, falls $\partial(Y) \subseteq T$.

Anmerkung: Eine Menge Y kann gleichzeitig offen und abgeschlossen sein, wie z.B.:

$Y := P \cup T$. In diesem Fall hängt es von der Interpretation bzw. Anwendung ab, ob Y durch einen Platz oder eine Transition ersetzt wird.



Elementare Vergrößerung



Definition: Sei $\mathcal{N} = (P, T, F)$ ein Netz und Y eine nicht leere Transitionsberandete Menge von Elementen.

Dann heißt $\mathcal{N}[Y] = (P[Y], T[Y], F[Y])$ *elementare Vergrößerung* von \mathcal{N} in Bezug auf Y , falls gilt:

1. $P[Y] = P \setminus Y$.
2. $T[Y] = (T \setminus Y) \cup \{t_Y\}$, wobei t_Y ein neues Element ist.
3. $F[Y] = \{(x, y) \mid x \notin Y \wedge y \notin Y \wedge (x, y) \in F\} \cup \{(x, t_Y) \mid x \notin Y \wedge \exists y \in Y . (x, y) \in F\} \cup \{(t_Y, x) \mid x \notin Y \wedge \exists y \in Y . (y, x) \in F\}$.

Wenn Y eine Platz-berandete Menge ist, dann erhält man entsprechend $\mathcal{N}[Y] = (P[Y], T[Y], F[Y])$ durch

1. $P[Y] = (P \setminus Y) \cup \{p_Y\}$, wobei p_Y ein neues Element ist,
2. $T[Y] = T \setminus Y$,
3. $F[Y] = \{(x, y) \mid x \notin Y \wedge y \notin Y \wedge (x, y) \in F\} \cup \{(x, p_Y) \mid x \notin Y \wedge \exists y \in Y . (x, y) \in F\} \cup \{(p_Y, x) \mid x \notin Y \wedge \exists y \in Y . (y, x) \in F\}$.

Vergrößerung/Verfeinerung

Iteration der Vergrößerung:

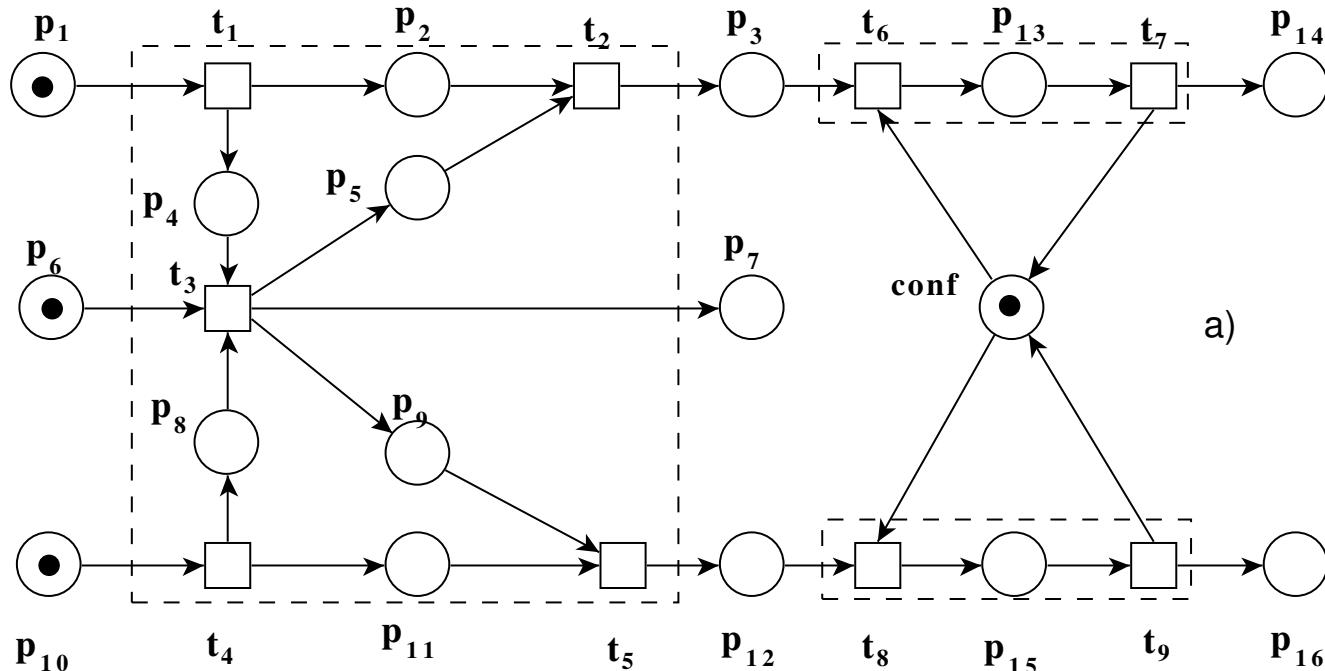
$$\mathcal{N}_2 = \mathcal{N}_1[Y_1, Y_2, \dots, Y_n]$$

\mathcal{N}_2 heißt **Vergrößerung** von \mathcal{N}_1

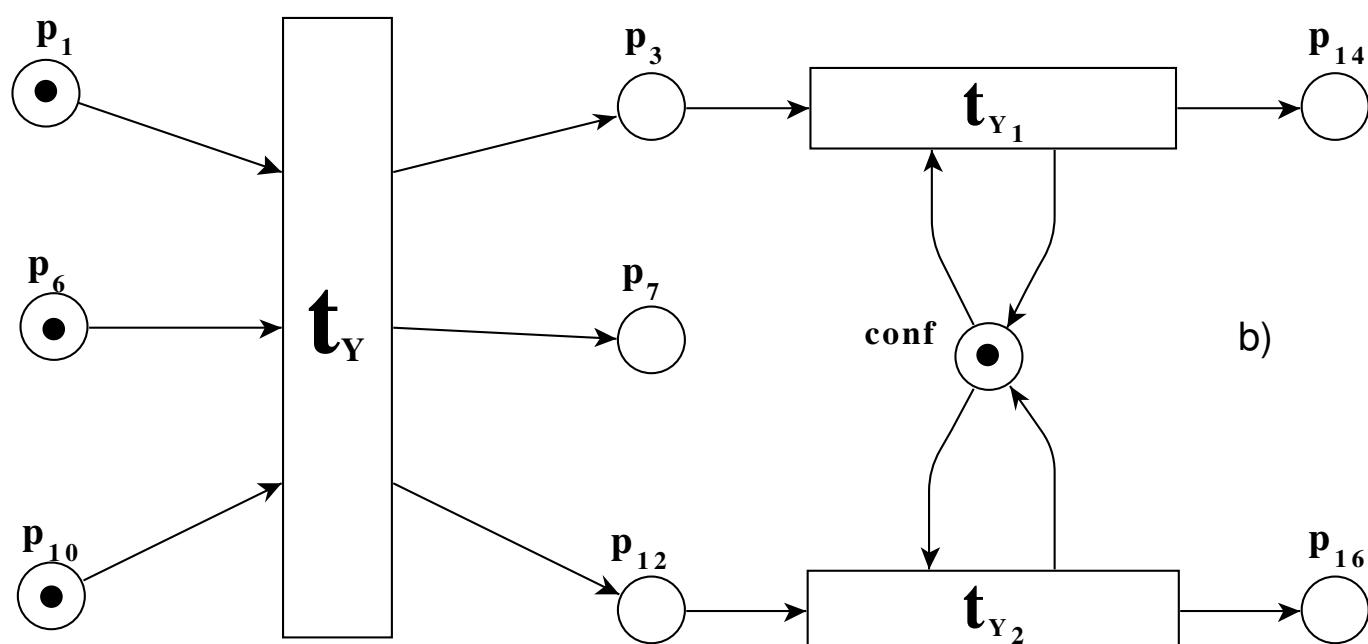
\mathcal{N}_1 heißt **Verfeinerung** von \mathcal{N}_2

$n = 1$: einfache

Beispiel

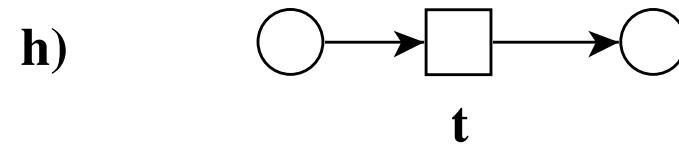
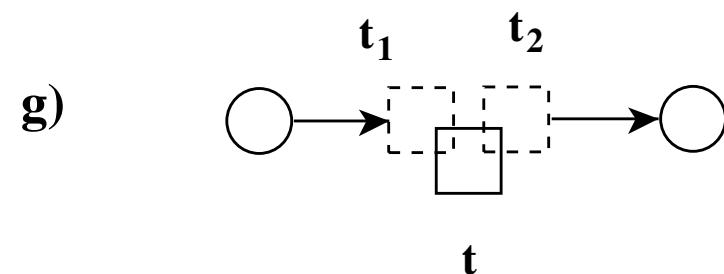
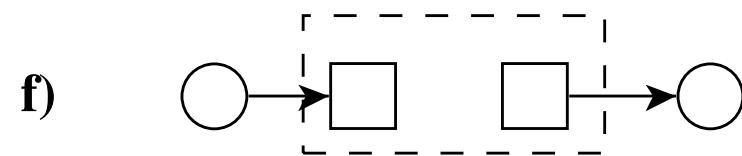
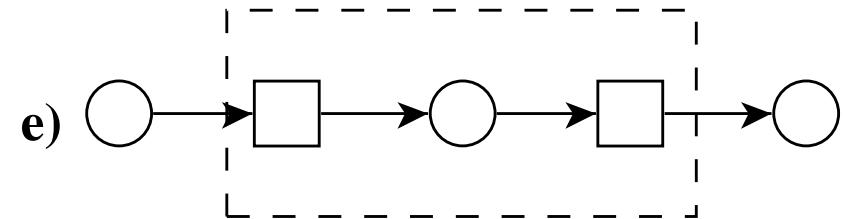
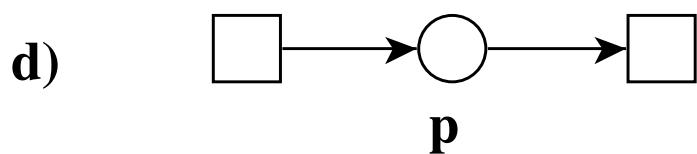
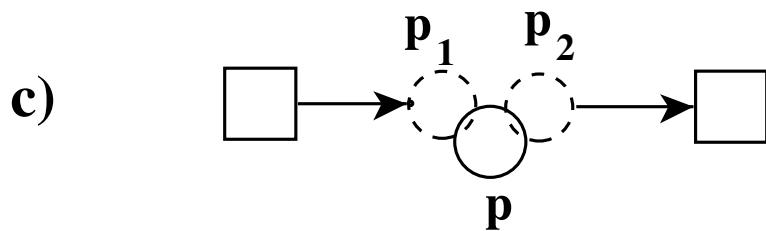
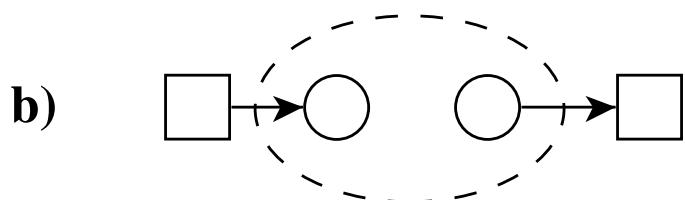
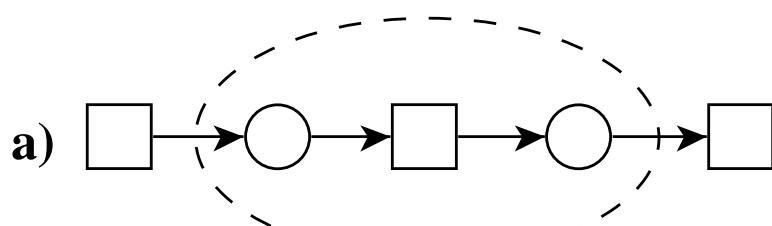


a)

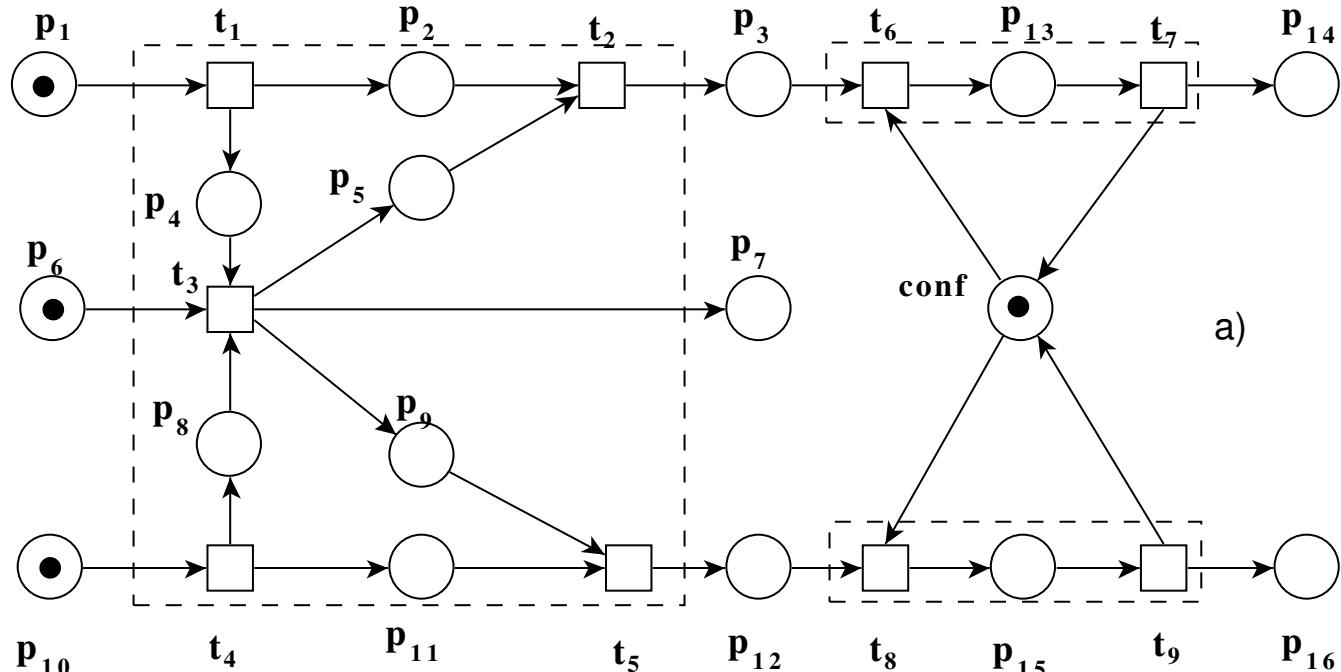
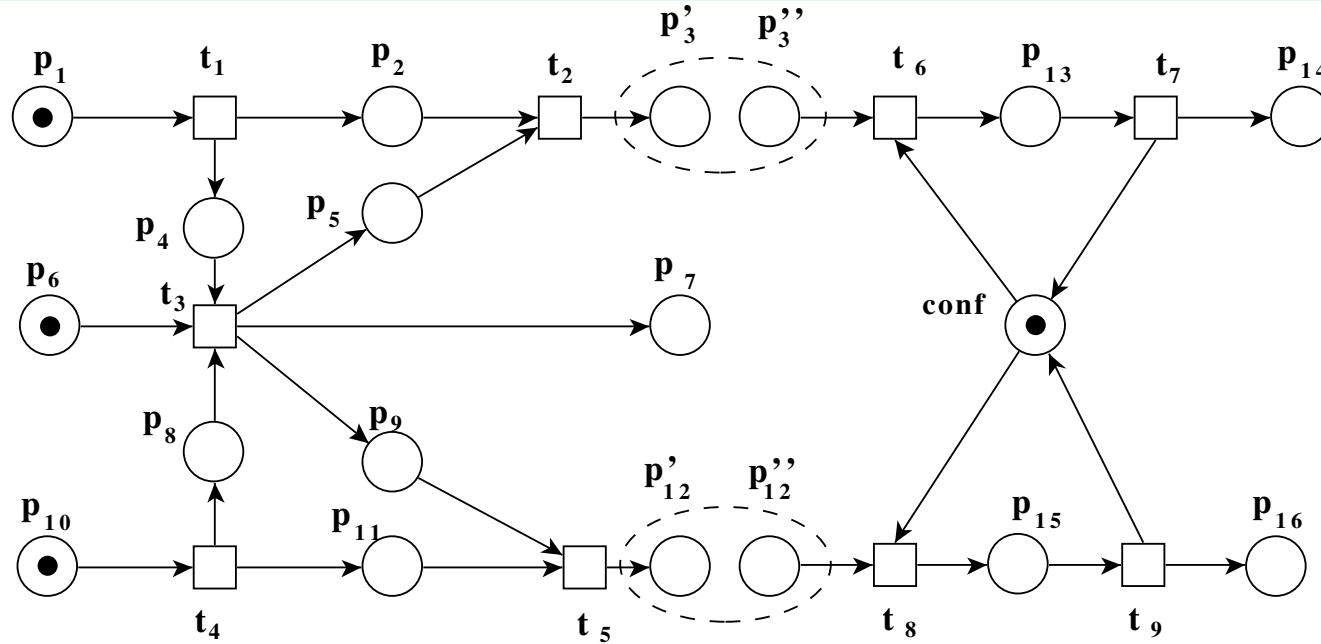


b)

Vergrößerung vs. Verschmelzung



Beispiel



Netzmorphismen

Unter Strukturverträglichkeit ist zu verstehen, dass bei einer Abbildung ϕ eines Netzes $\mathcal{N}_1 = (P_1, T_1, F_1)$ auf $\mathcal{N}_2 = (P_2, T_2, F_2)$ die Kanten F „zusammen“ mit den Plätzen F -erhaltend abgebildet wird:

$$\forall x, y \in (P_1 \cup T_1) : (x, y) \in F_1 \Rightarrow ((\phi(x), \phi(y)) \in F_2 \vee \phi(x) = \phi(y))$$

Mit anderen Worten, die Abbildung der Kanten ergibt sich aus der Abbildung der Knoten.

Eine Abbildung ϕ kann die Kantenrichtung umdrehen, indem einer Kante $(p, t) \in F_1$ die Kante $(\phi(p), \phi(t)) \in F_2$ zugeordnet wird, bei der $\phi(p)$ eine Transition und $\phi(t)$ ein Platz ist. Für Netzmorphismen wird ein solches Umdrehen der Richtung verboten.

Wir definieren für Kanten $(x, y) \in F$ die Notation $\phi(x, y) := (\phi(x), \phi(y))$.

Definition: Seien $\mathcal{N}_1 = (P_1, T_1, F_1)$ und $\mathcal{N}_2 = (P_2, T_2, F_2)$ zwei Netze.

- Eine Abbildung $\phi : (P_1 \cup T_1) \rightarrow (P_2 \cup T_2)$ heißt *Netzmorphismus*, falls gilt:

$$\begin{aligned} (x, y) \in F_1 \cap (P_1 \times T_1) &\Rightarrow (\phi(x, y) \in F_2 \cap (P_2 \times T_2) \vee \phi(x) = \phi(y)) \\ (x, y) \in F_1 \cap (T_1 \times P_1) &\Rightarrow (\phi(x, y) \in F_2 \cap (T_2 \times P_2) \vee \phi(x) = \phi(y)) \end{aligned}$$

- Ein Netzmorphismus ϕ heißt *Faltung*, falls $\phi(P_1) \subseteq P_2$ und $\phi(T_1) \subseteq T_2$ gilt.

Epimorphismen

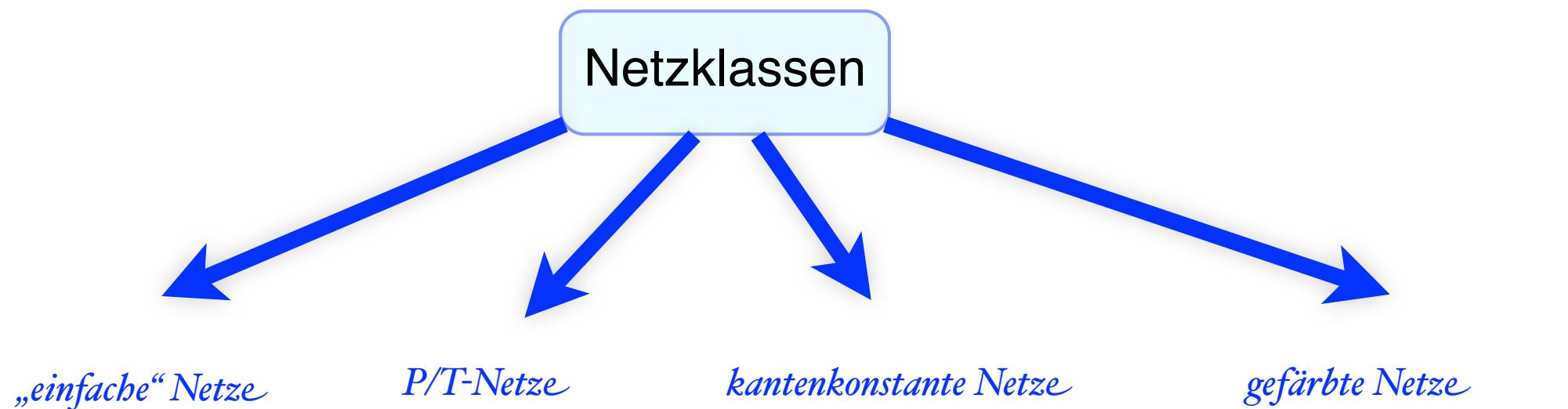
Ist der Netzmorphismus ϕ surjektiv, so sind die Bildknoten $P_2 \cup T_2$ aus den Konoten des Netzes \mathcal{N}_1 mit ϕ konstruierbar. Gilt zudem noch, dass jede Kante in \mathcal{N}_2 ein Abbild einer Kante in \mathcal{N}_1 ist, dann wird das Netz \mathcal{N}_2 komplett durch \mathcal{N}_1 und ϕ konstruierbar. Solche Morphismen heißen Epimorphismen.

- Ein Netzmorphismus ϕ ist ein *Epimorphismus*, falls ϕ surjektiv ist und für jede Kante ein Urbild existiert, d.h. für alle $f_2 \in F_2$ existiert ein $f_1 \in F_1$ mit $\phi(f_1) = f_2$
Eine Faltung mit dieser Eigenschaft heißt *Epifaltung*.
- Ein Netzmorphismus ϕ ist ein *Netzisomorphismus*, falls ϕ eine Bijektion ist und ϕ^{-1} ebenfalls ein Netzmorphismus ist.

Theorem: Seien \mathcal{N}_1 und \mathcal{N}_2 zwei endliche Netze.

- \mathcal{N}_2 ist genau dann eine Vergrößerung von \mathcal{N}_1 , wenn es einen Epimorphismus von \mathcal{N}_1 nach \mathcal{N}_2 gibt.

Netzklassen

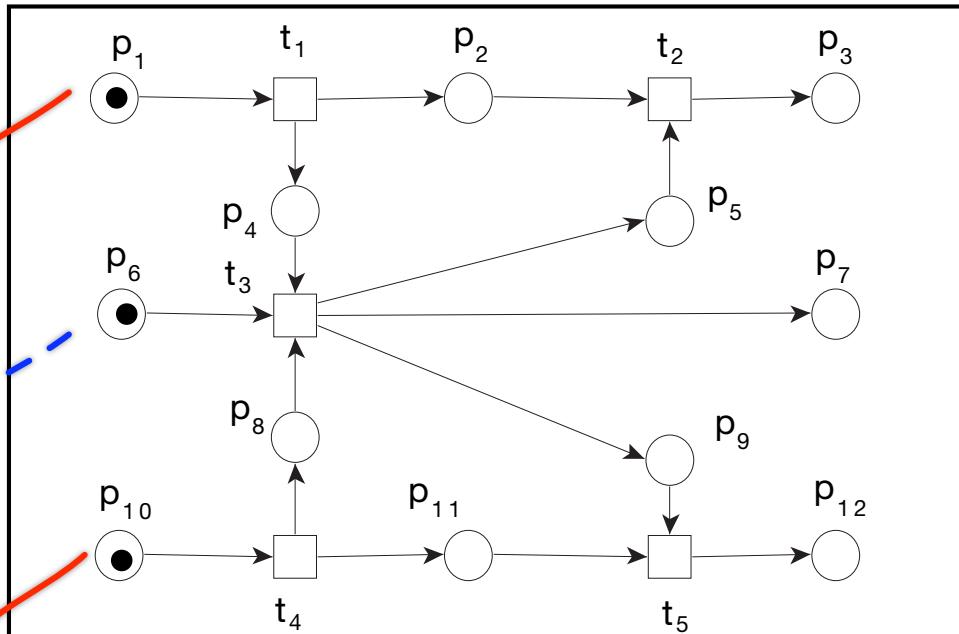


FGI 2

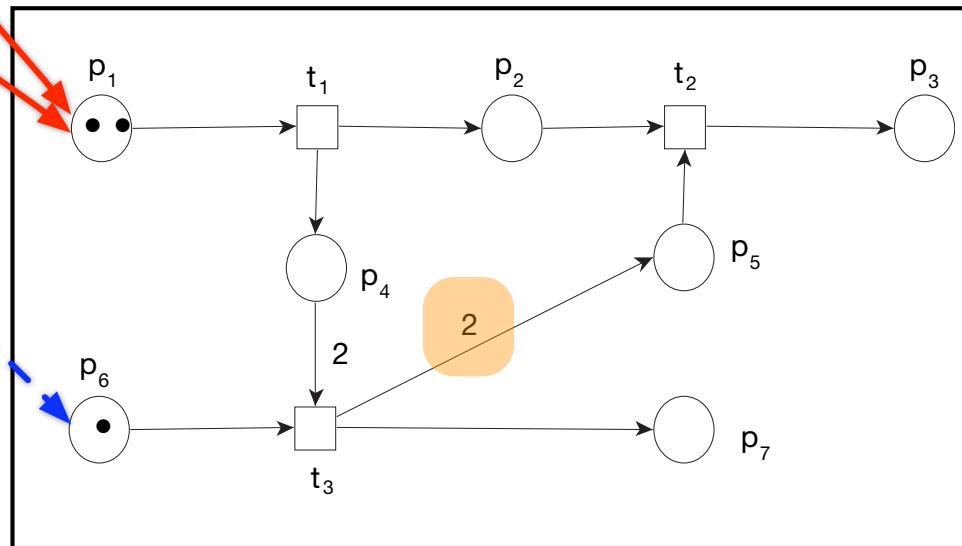
Daniel Moldt

P/T-Netze

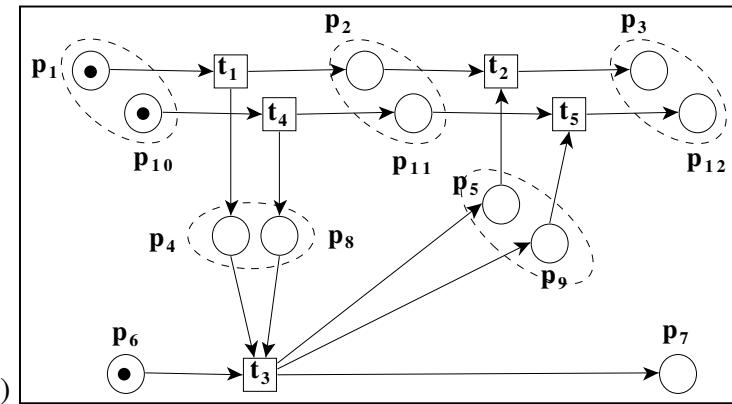
Platz/Transitions-Netze



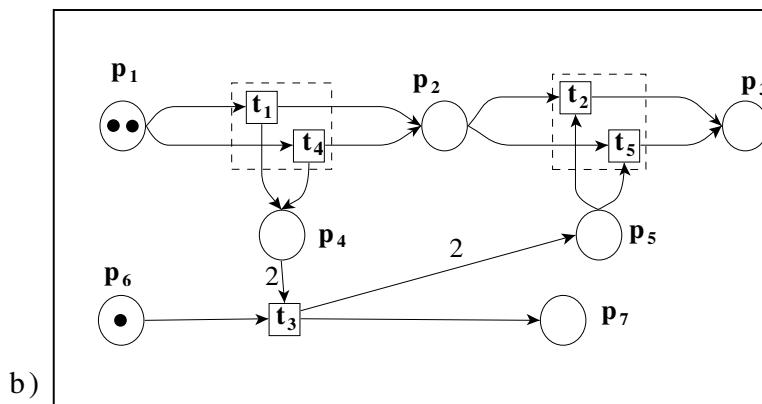
neu:
Mehrfachmarkierung +
Kantengewichte



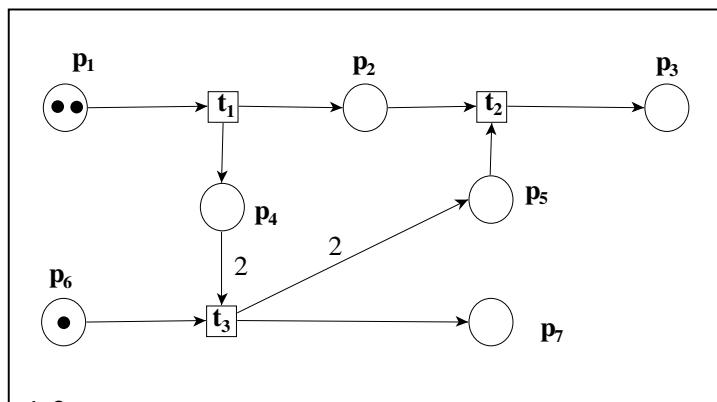
Faltungen



Φ_1



Φ_2



P/T-Netze

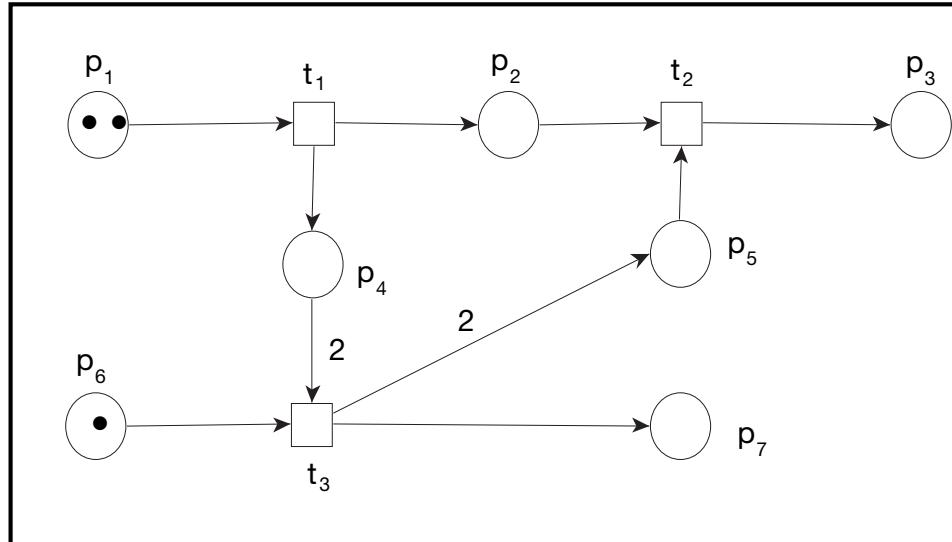


Abbildung 3.12: Platz/Transitions Netz \mathcal{N}_3

Definition: Ein *Platz/Transitions-Netz* $\mathcal{N} = \langle P, T, F, W, \mathbf{m}_0 \rangle$ besteht aus den folgenden Komponenten:

- (P, T, F) ist ein endliches Netz.
- $W : F \rightarrow \mathbb{N} \setminus \{0\}$ ist die Kantengewichtung. z.B.: $W(t_3, p_5) = 2$
- $\mathbf{m}_0 : P \rightarrow \mathbb{N}$ ist die Anfangsmarkierung.

Ein P/T-Netz heißt *einfaches Netz*, falls $W(x, y) = 1$ für alle $(x, y) \in F$ gilt.

Darstellung von \mathbf{m}_0 ist auch als Vektor üblich.

Markierungen

Darstellung als Abbildung:

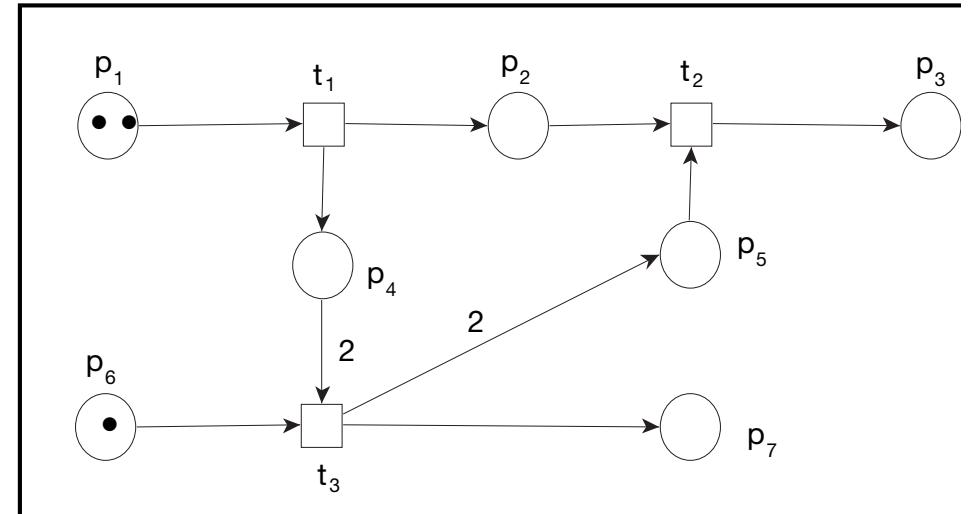
$$\mathbf{m}(p_1) = 2, \mathbf{m}(p_6) = 1, \mathbf{m}(p_i) = 0 \text{ für } i \in \{2, 3, 4, 5, 7\}$$

Darstellung als Vektor:

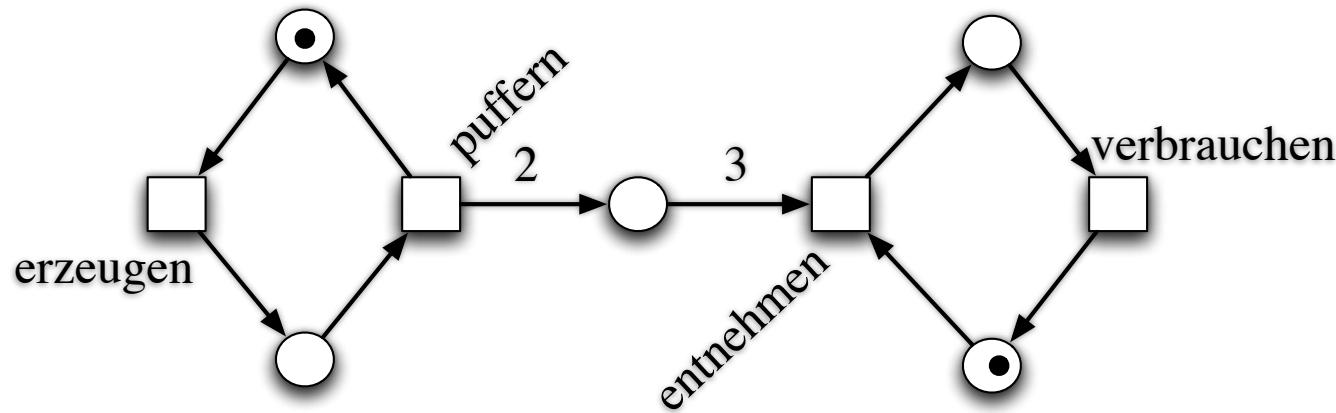
$$\begin{matrix} p_1 \\ \vdots \\ p_7 \end{matrix} \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Darstellung als Wort:

$$\mathbf{m}_0 = p_1^2 p_6$$



Erzeuger-Verbraucher



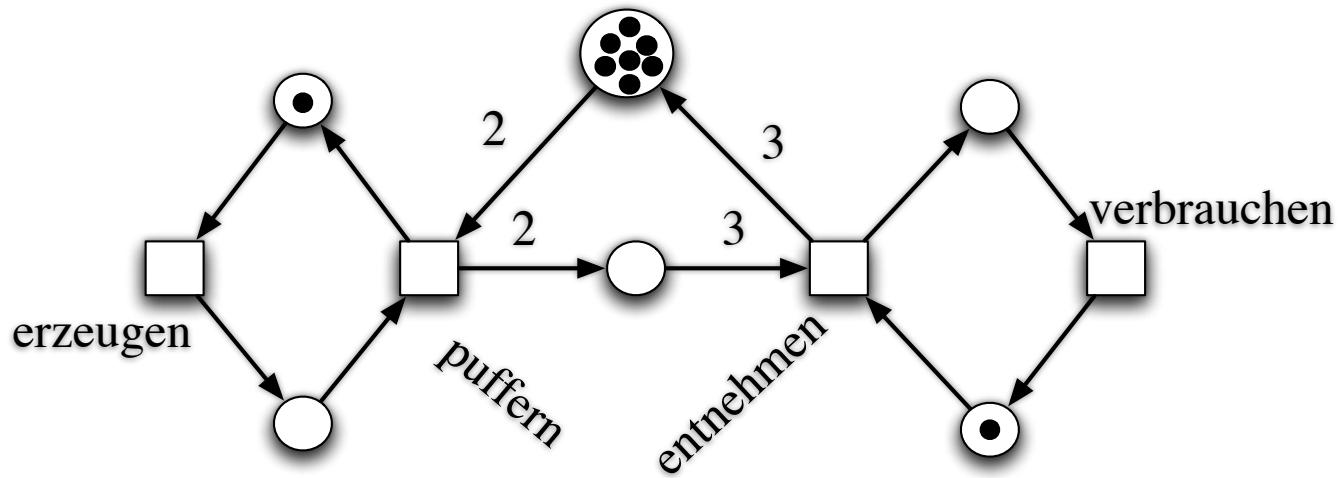
Erzeuger - Verbraucher - System

Sender - Empfänger - System

unbeschränkter Puffer

Erzeuger-Verbraucher mit Puffer

Komplementäre Stelle

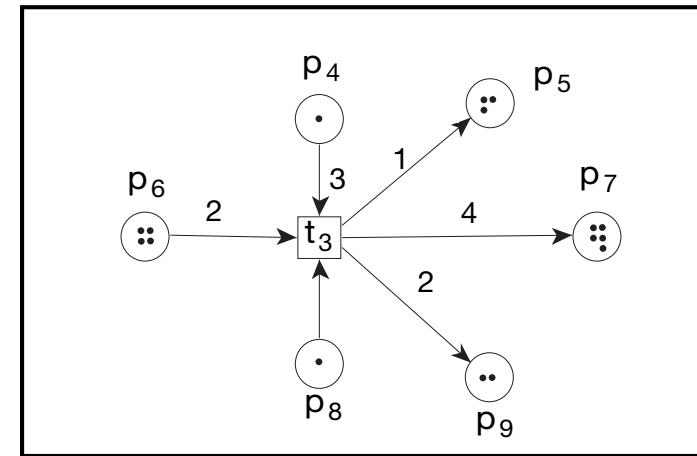
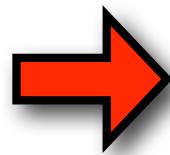
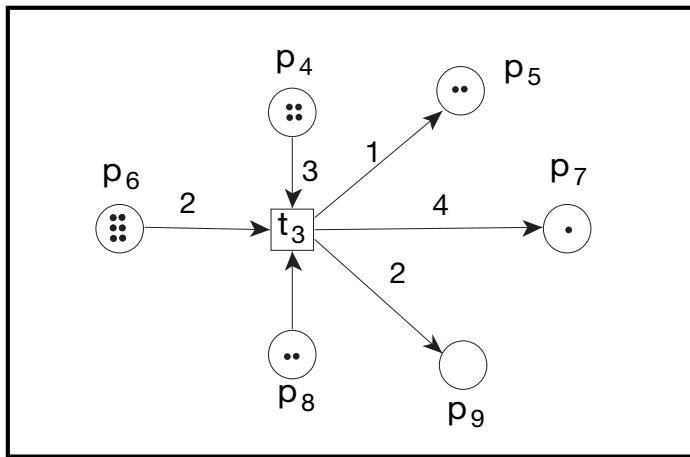


Erzeuger - Verbraucher - System

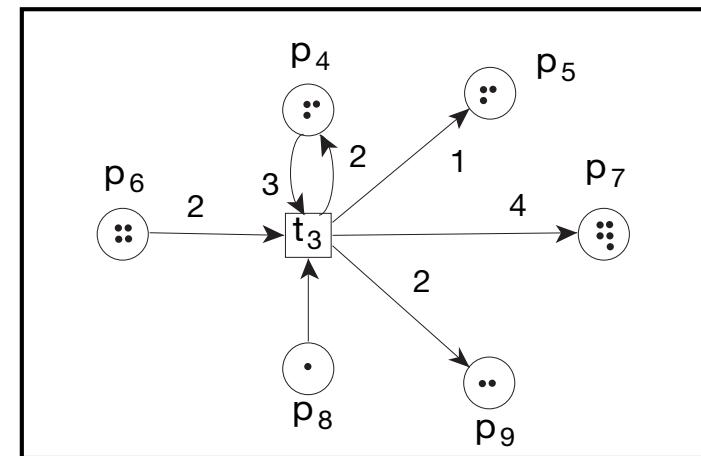
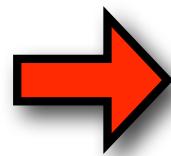
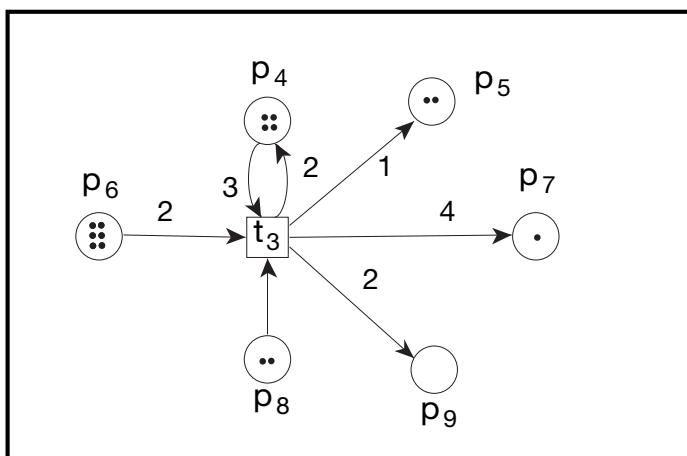
Sender - Empfänger - System

beschränkter Puffer

Schalten



Nebenbedingung



Aktivierung von Transitionen

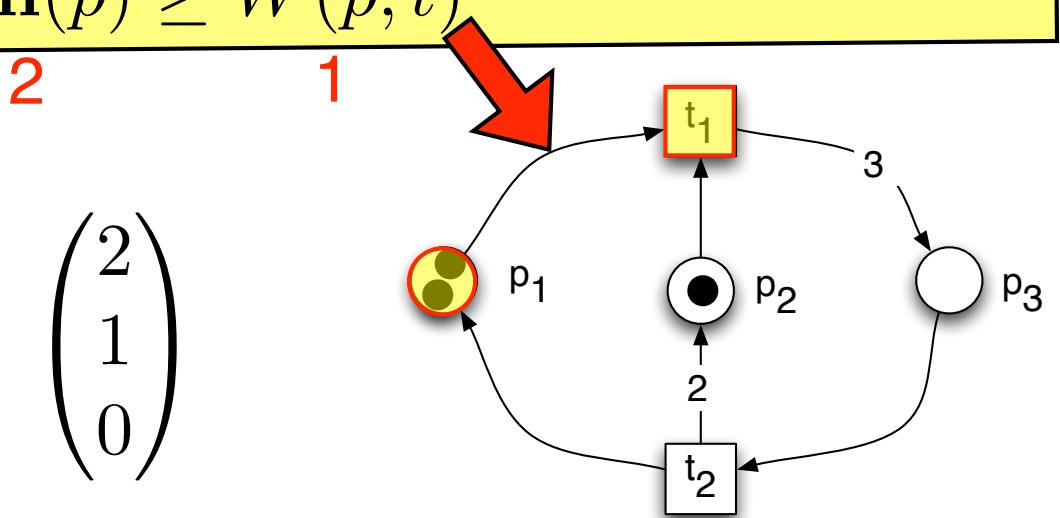
Definition 5.2 a) Die **Markierung** eines P/T – Netzes $\mathcal{N} = \langle P, T, F, W, \mathbf{m}_0 \rangle$ ist ein Vektor \mathbf{m} mit $\mathbf{m}(p) \in \mathbb{N}$ für jedes $p \in P$ (auch als Abbildung $\mathbf{m} : P \rightarrow \mathbb{N}$ aufzufassen). Die Menge aller Markierungen über P (bzw. S) wird mit M_P (bzw. M_S) bezeichnet.

b) Eine Transition $t \in T$ heißt **aktiviert** in einer Markierung \mathbf{m} falls

$$\forall p \in {}^{\bullet}t. \quad \mathbf{m}(p) \geq W(p, t)$$

(als Relation: $\mathbf{m} \xrightarrow{t}$).

Markierung $\mathbf{m} =$
$$\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$$



Nachfolgemarkierung

c) Es sei $\widetilde{W}(p, t) := \begin{cases} W(p, t) & \text{falls } (p, t) \in F \\ 0 & \text{sonst} \end{cases}$

und entsprechend

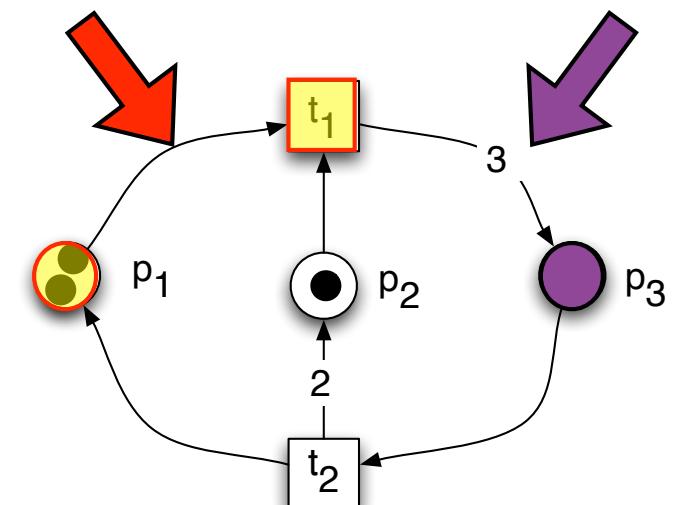
$\widetilde{W}(t, p) := \begin{cases} W(t, p) & \text{falls } (t, p) \in F \\ 0 & \text{sonst} \end{cases}$

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

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

$$p = p_1, t = t_1 \quad 2 \quad 1 \quad 1 \quad 2 \quad 1 \quad 0$$

$$p = p_3, t = t_1 \quad 0 \quad 0 \quad 3 \quad 0 \quad 0 \quad 3$$

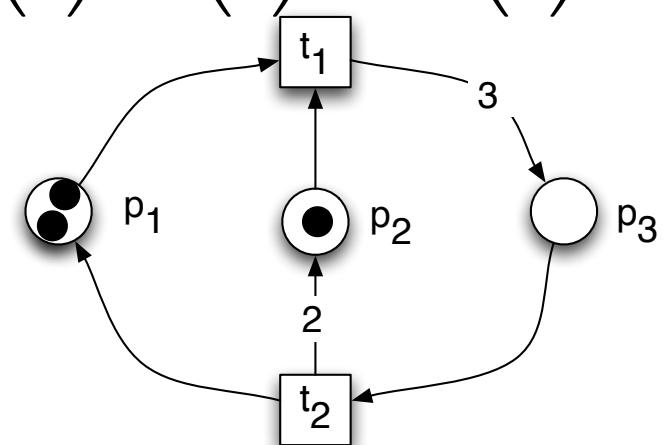


Vektor-Notation

d) Definiert man $\mathbf{W}(\bullet, t) := (\widetilde{W}(p_1, t), \dots, \widetilde{W}(p_{|P|}, t))$ als Vektor der Länge $|P|$ und entsprechend $\mathbf{W}(t, \bullet) := (\widetilde{W}(t, p_1), \dots, \widetilde{W}(t, p_{|P|}))$, dann kann die Nachfolgemarkierung einfacher durch Vektoren definiert werden:

$$\mathbf{m} \xrightarrow{t} \mathbf{m}' \Leftrightarrow \mathbf{m} \geq \mathbf{W}(\bullet, t) \wedge \mathbf{m}' = \mathbf{m} - \mathbf{W}(\bullet, t) + \mathbf{W}(t, \bullet))$$

$$\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix} \quad \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix}$$



Schaltfolgen

Die Nachfolgemarkierungsrelation wird auf Wörter über T erweitert:

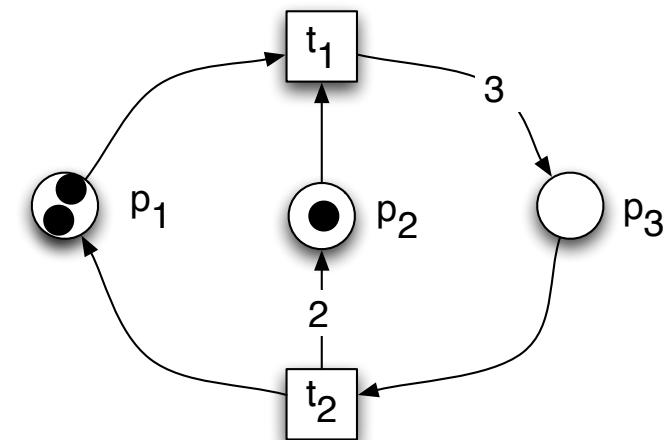
Definition: Sei \mathcal{N} ein P/T-Netz.

- $\mathbf{m} \xrightarrow{w} \mathbf{m}'$ falls w das leere Wort λ ist und $\mathbf{m} = \mathbf{m}'$,
- $\mathbf{m} \xrightarrow{wt} \mathbf{m}'$ falls $\exists \mathbf{m}''' : \mathbf{m} \xrightarrow{w} \mathbf{m}''' \wedge \mathbf{m}''' \xrightarrow{t} \mathbf{m}'$ für $w \in T^*$ und $t \in T$.

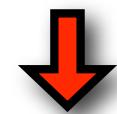
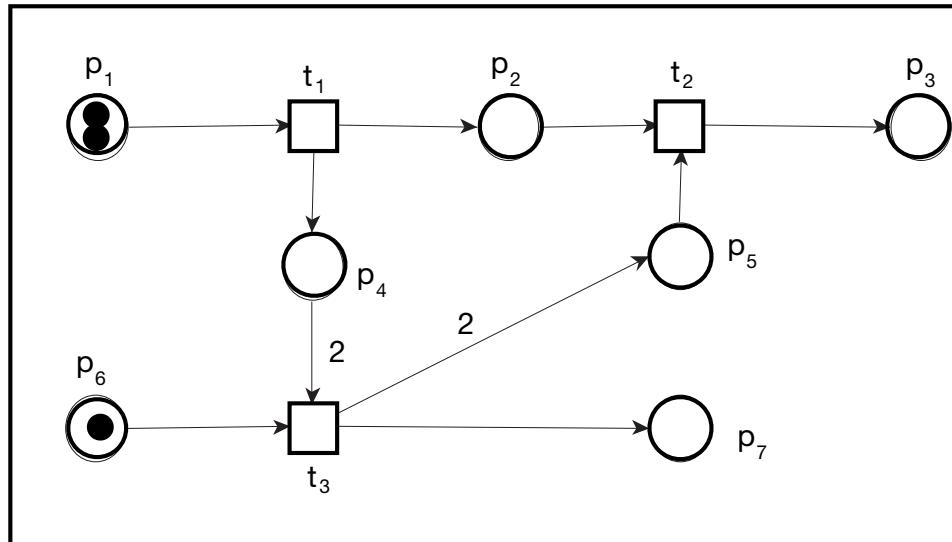
Eine Transitionsfolge $w \in T^*$ heißt *aktiviert* in \mathbf{m} (in Zeichen: $\mathbf{m} \xrightarrow{w}$), falls $\exists \mathbf{m}_1 : \mathbf{m} \xrightarrow{w} \mathbf{m}_1$

$FS(\mathcal{N}) := \{w \in T^* \mid \mathbf{m}_0 \xrightarrow{w}\}$ ist die Menge der *Schaltfolgen* (firing sequence set) von \mathcal{N} .

$$\begin{array}{ccc} \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} & \xrightarrow{\lambda} & \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} & \xrightarrow{t_1 t_2 t_1 t_2 t_1} & \begin{pmatrix} 1 \\ 2 \\ 7 \end{pmatrix} \end{array}$$



Beispiel



$$\begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{t_1} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{t_1} \begin{pmatrix} 0 \\ 2 \\ 0 \\ 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{t_3} \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 2 \\ 0 \\ 1 \end{pmatrix} \xrightarrow{t_2} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \xrightarrow{t_2} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Monotonie der Schaltregel

Lemma: Für jedes P/T Netz gilt die folgende Additivität:

$$\forall w \in T^* : \mathbf{m}_1 \xrightarrow{w} \mathbf{m}_3 \Rightarrow \forall \mathbf{m} : (\mathbf{m}_1 + \mathbf{m}) \xrightarrow{w} (\mathbf{m}_3 + \mathbf{m}) \quad (1.1)$$

(Da die Multimengenaddition kommutativ ist, könnten wir \mathbf{m} genausogut von links addieren.)

Beweis: Beweis der Behauptung (1.1) per Induktion über die Länge von w

Lemma: Für jedes P/T-Netz gilt die folgende Monotonie-Eigenschaft:

$$(\mathbf{m}_1 \xrightarrow{w} \mathbf{m}_3 \wedge \mathbf{m}_2 \geq \mathbf{m}_1) \Rightarrow \mathbf{m}_2 \xrightarrow{w} \mathbf{m}_4 \quad (1.2)$$

Beweis:

Setzen wir in (1.1) speziell $\mathbf{m} := \mathbf{m}_2 - \mathbf{m}_1 \geq \mathbf{0}$, so ergibt sich (1.2):

$$\mathbf{m}_1 \xrightarrow{w} \mathbf{m}_3 \Rightarrow \mathbf{m}_2 = (\mathbf{m}_1 + \mathbf{m}) \xrightarrow{w} (\mathbf{m}_3 + \mathbf{m}) = (\mathbf{m}_3 + \mathbf{m}_2 - \mathbf{m}_1) = \mathbf{m}_4$$

D.h. die Folgemarkierung ergibt sich als $\mathbf{m}_4 = (\mathbf{m}_3 + \mathbf{m}_2 - \mathbf{m}_1)$. □

Erreichbare Markierungen

Sei \mathcal{N} ein P/T-Netz. Für eine Menge S^0 von Markierungen sei dann die Menge der von S^0 aus *erreichbaren* Markierungen definiert als:

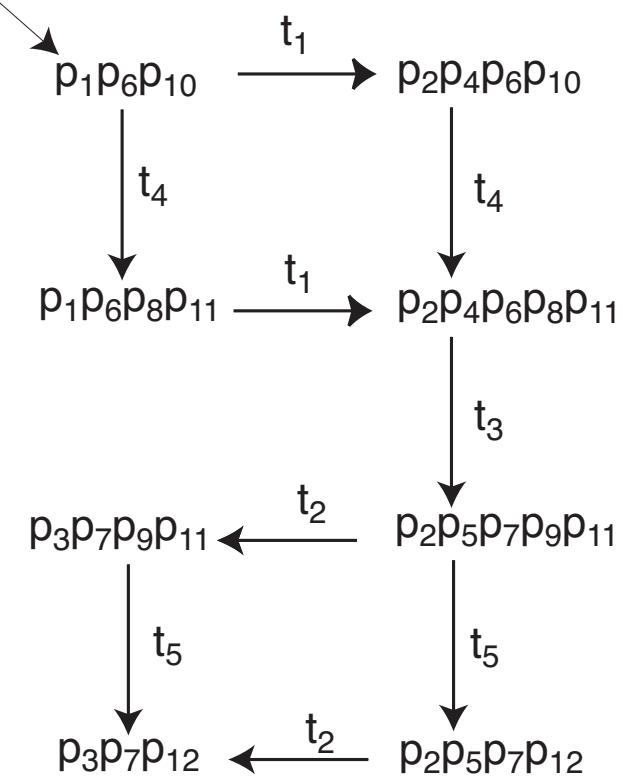
$$\mathbf{R}(\mathcal{N}, S^0) := \{\mathbf{m}_2 \mid \exists w \in T^*, \mathbf{m}_1 \in S^0 : \mathbf{m}_1 \xrightarrow{w} \mathbf{m}_2\}$$

Die *Erreichbarkeitsmenge* $\mathbf{R}(\mathcal{N})$ ist $\mathbf{R}(\mathcal{N}) := \mathbf{R}(\mathcal{N}, \{\mathbf{m}_0\})$.

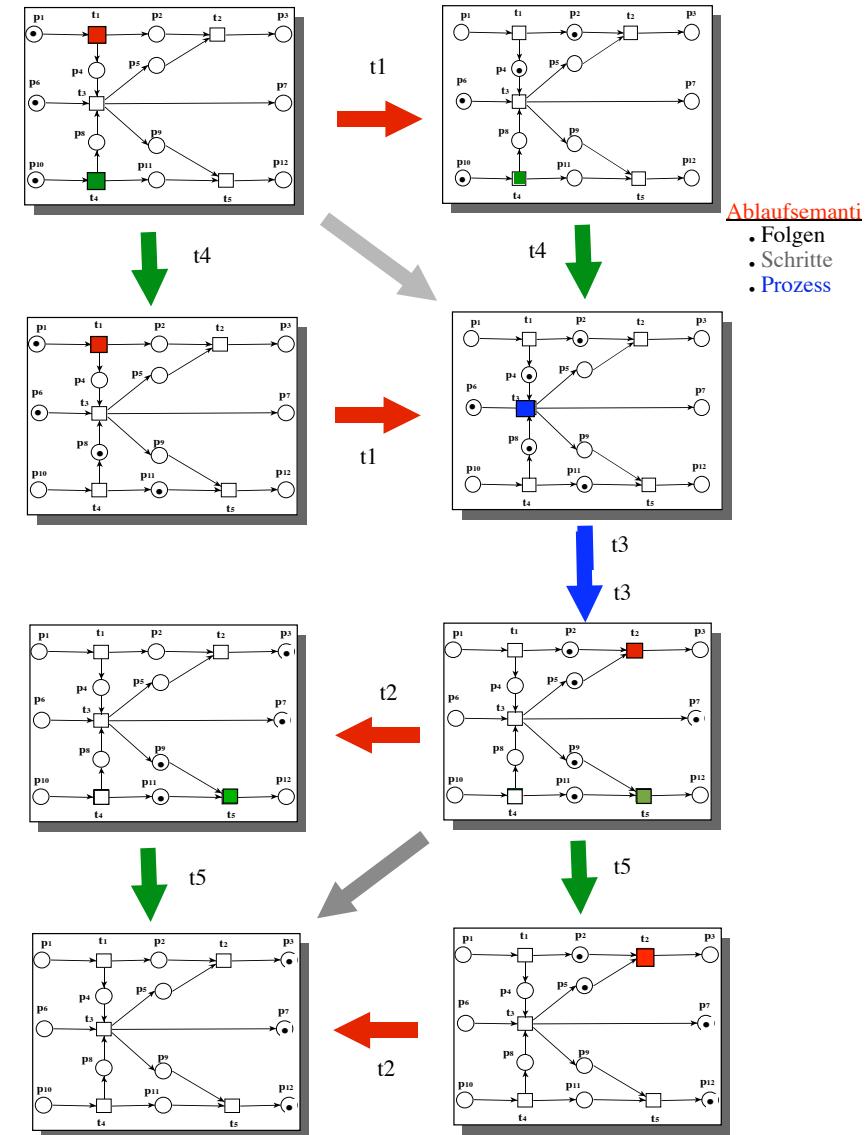
Falls \mathcal{N} implizit gegeben ist, schreiben wir auch $\mathbf{R}(\mathbf{m})$ für $\mathbf{R}(\mathcal{N}, \{\mathbf{m}\})$.

Definition: Der *Erreichbarkeitsgraph* eines P/T-Netzes \mathcal{N} ist ein Graph $RG(\mathcal{N}) := (Kn, Ka)$ mit Knotenmenge $Kn := \mathbf{R}(\mathcal{N})$ und Kantenmenge $Ka := \{(\mathbf{m}_1, t, \mathbf{m}_2) \mid \mathbf{m}_1 \xrightarrow{t} \mathbf{m}_2\}$.

Beispiel

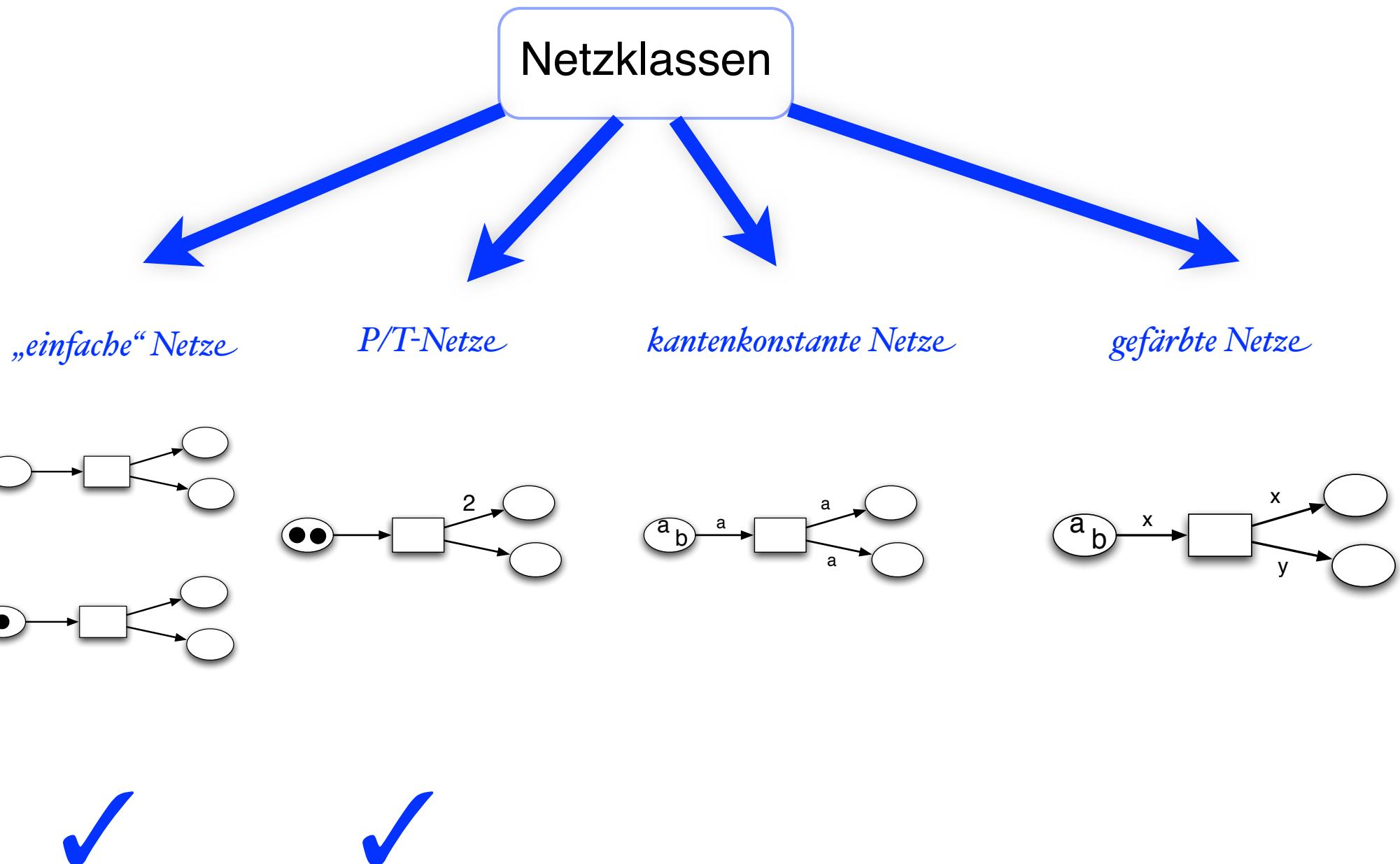


Transitionssystem



- Ablaufsemantik
- Folgen
 - Schritte
 - Prozess

Netzklassen



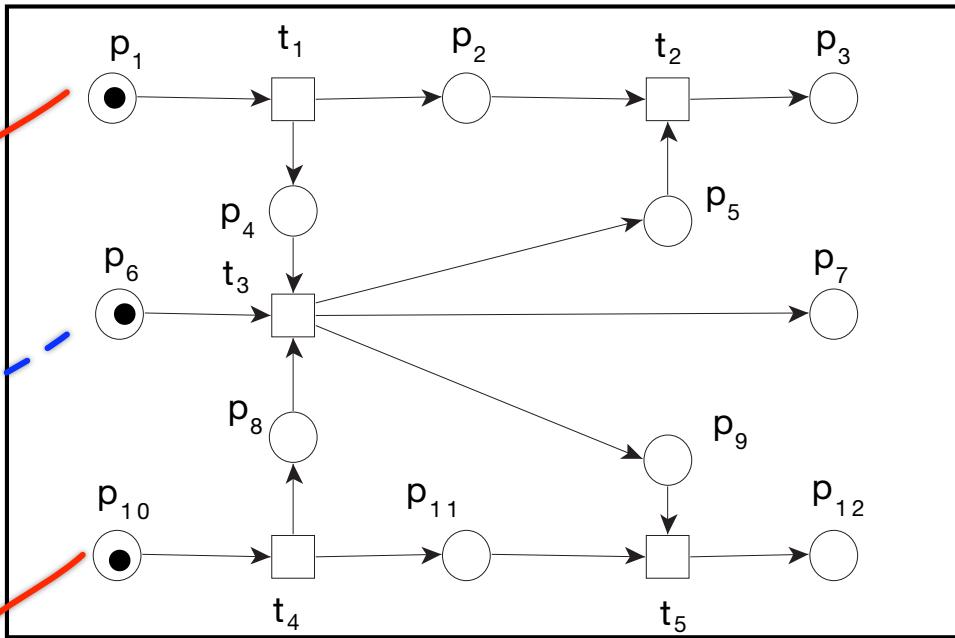
FGI 2

Daniel Moldt

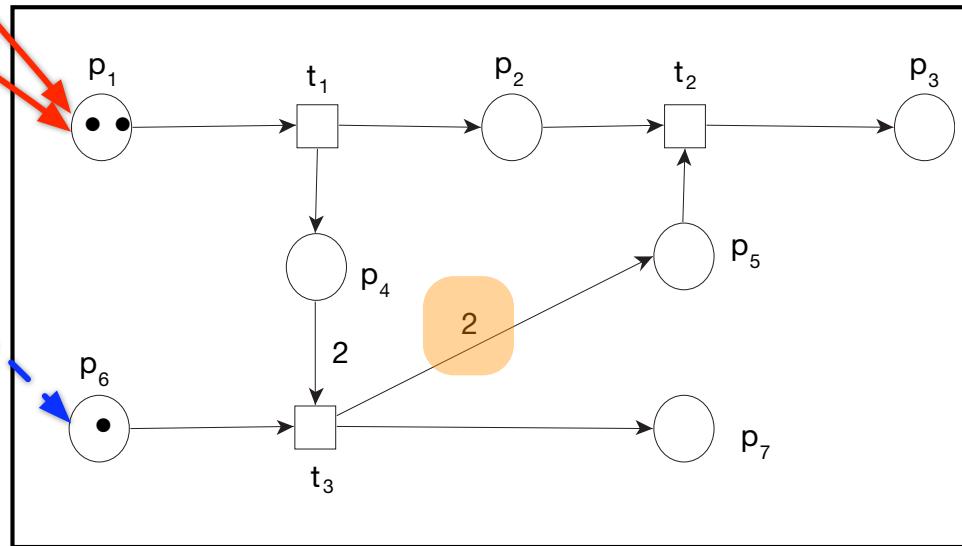
KKN

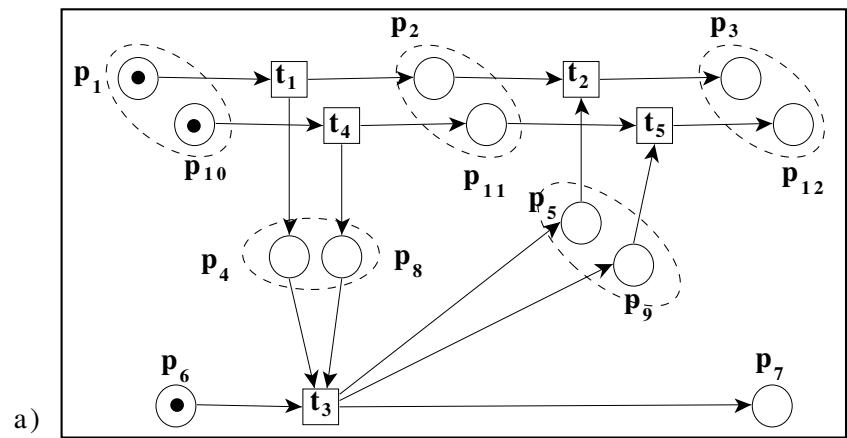
Ausblick

Platz/Transitions-Netze



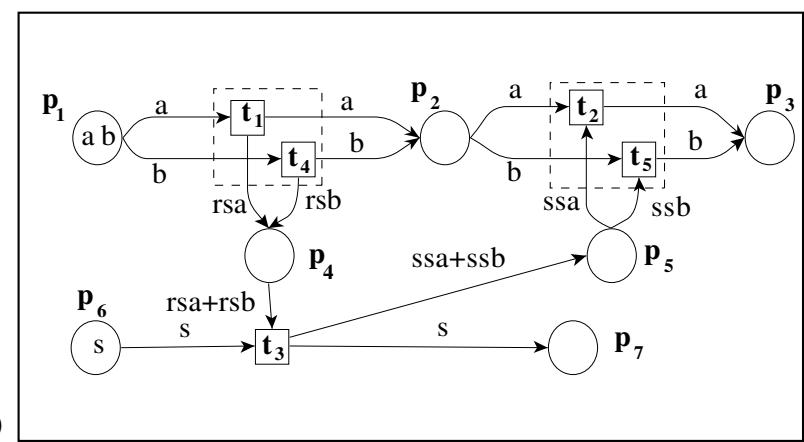
Mehrfachmarkierung +
Kantengewichte



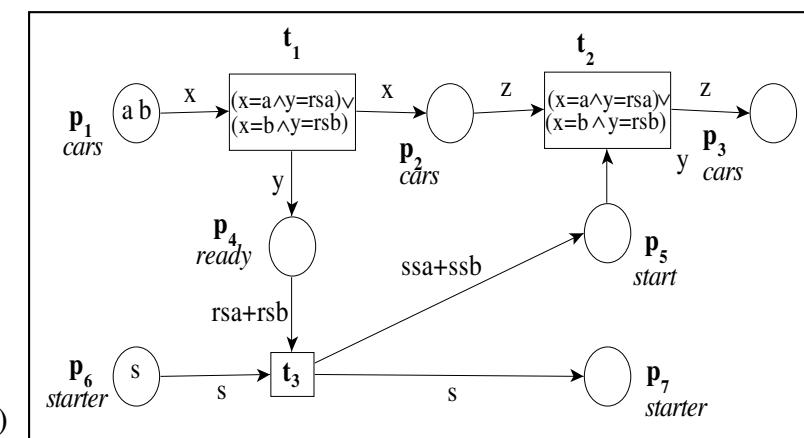


(Netzfaltung zum
kantenkonstanten Netz)

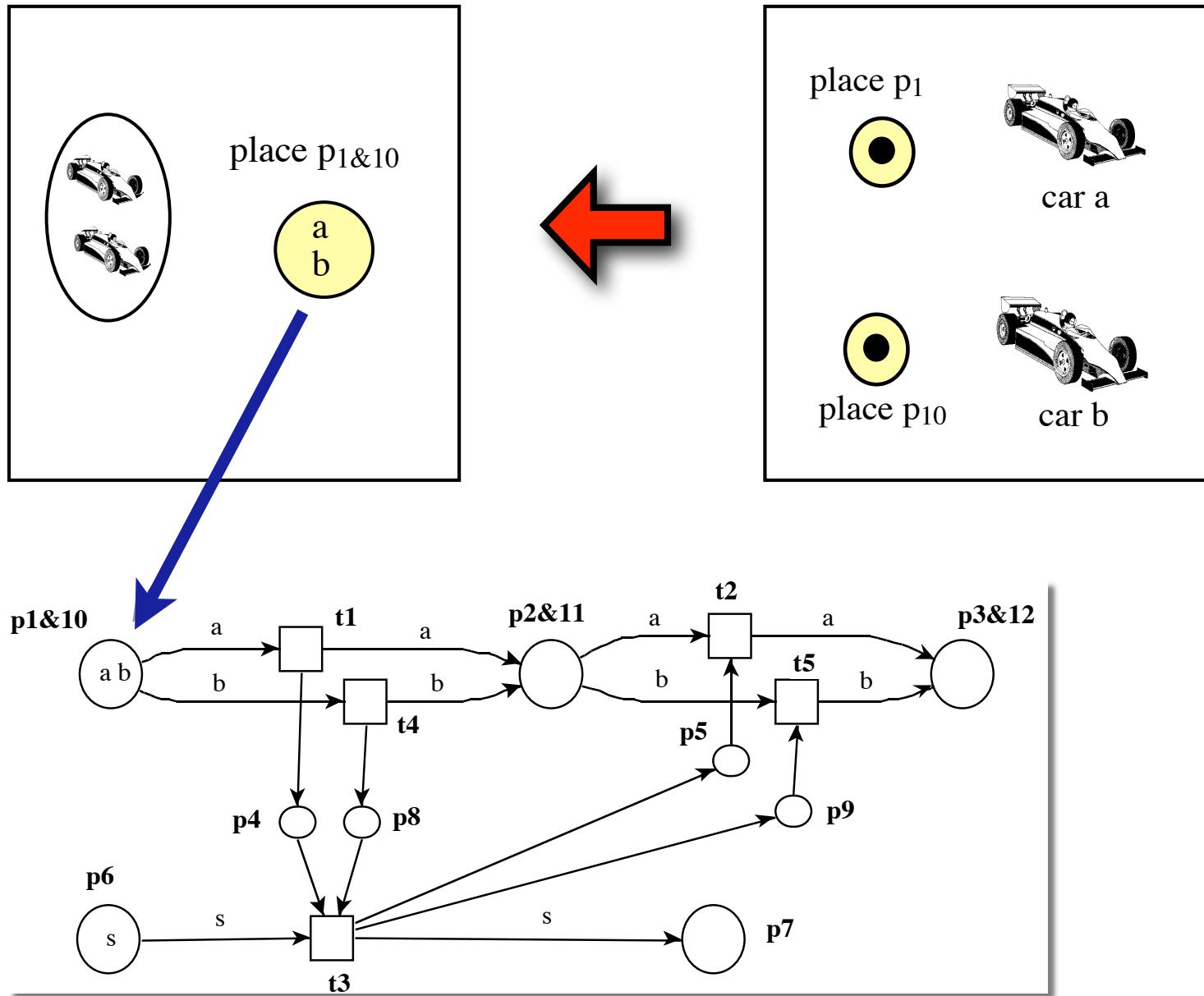
Φ_3

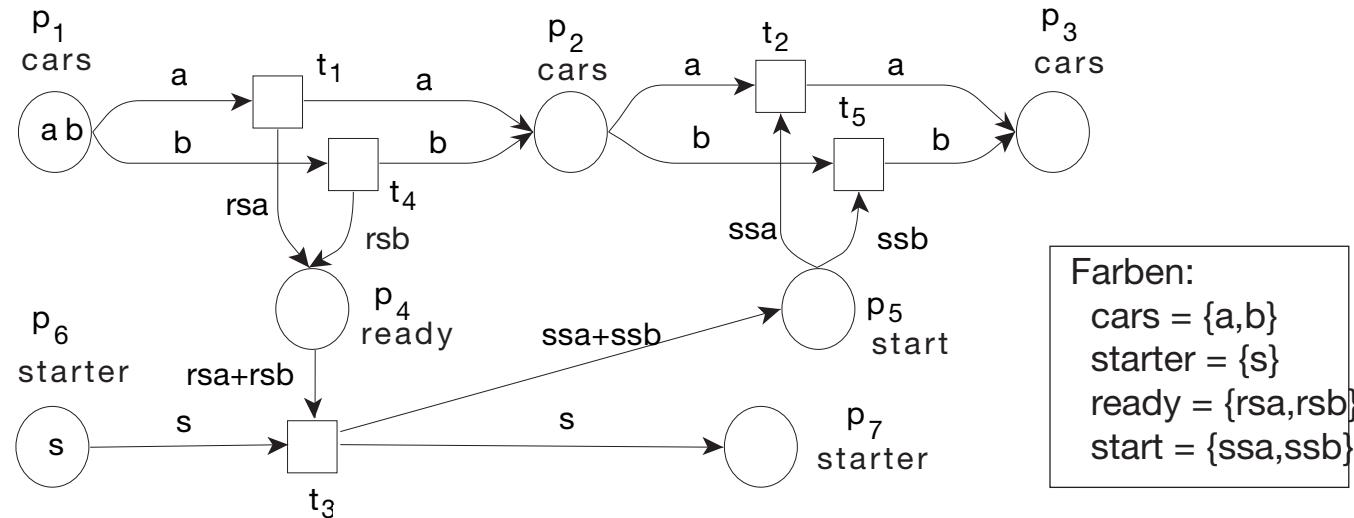


Φ_4



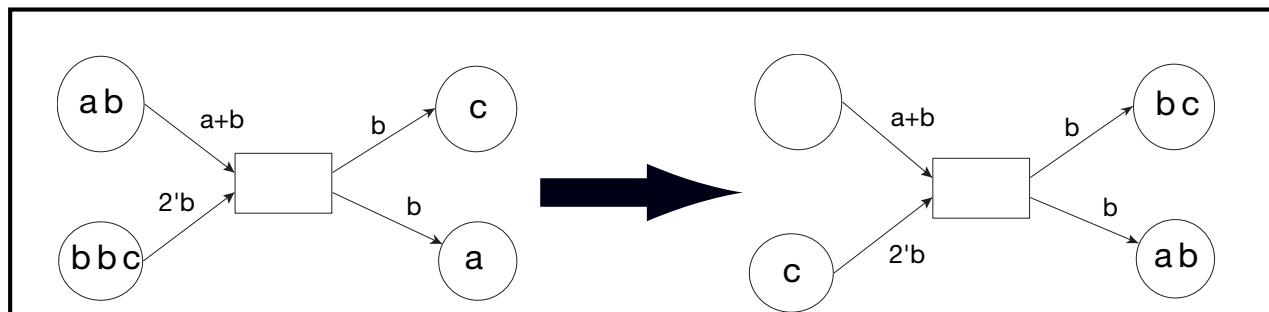
kantenkonstante Netze





2.10

Abbildung 3.10: Das kantenkonstante gefärbte Netz \mathcal{N}_2



Multimenge
"bag"

Abbildung 2.11 Schaltregel für kantenkonstante CPN

Markierungen in P/T-Netzen

Darstellung als Abbildung:

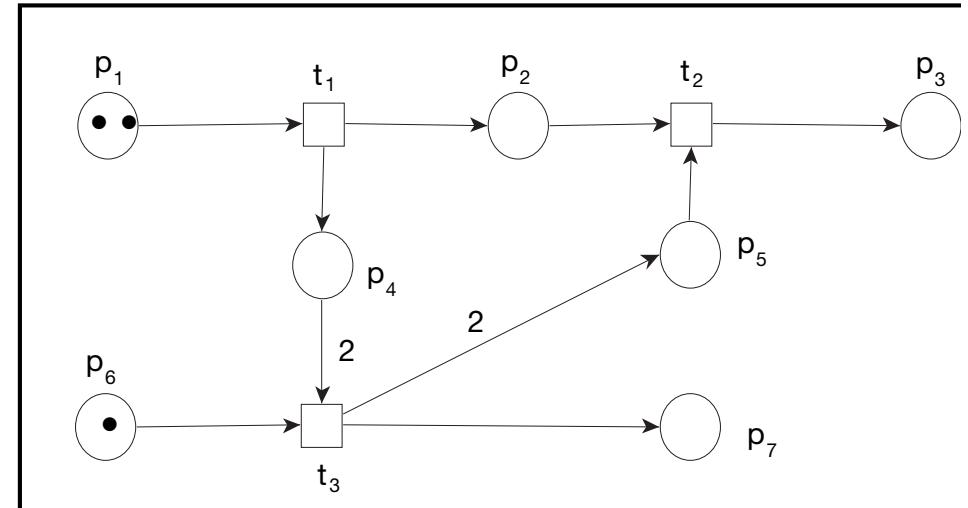
$$\mathbf{m}(p_1) = 2, \mathbf{m}(p_6) = 1, \mathbf{m}(p_i) = 0 \text{ für } i \in \{2, 3, 4, 5, 7\}$$

Darstellung als Vektor:

$$\begin{matrix} p_1 \\ \vdots \\ p_7 \end{matrix} \left(\begin{array}{c} 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{array} \right)$$

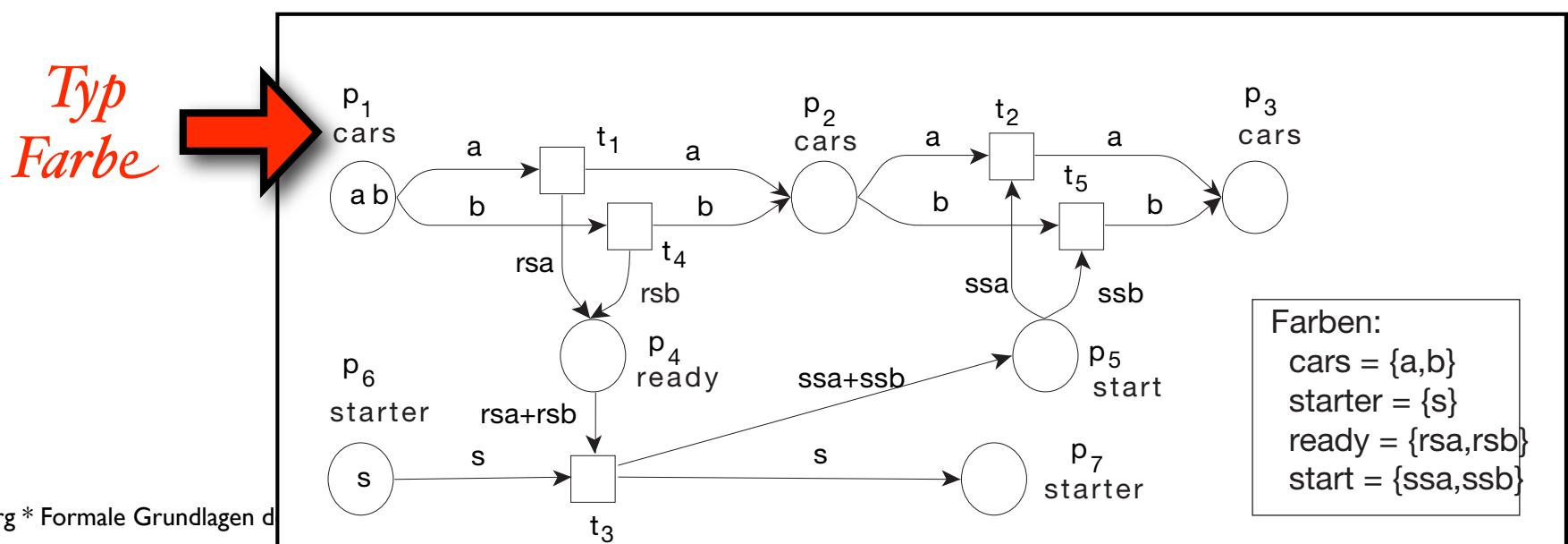
Darstellung als Wort:

$$\mathbf{m}_0 = p_1^2 p_6$$



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



Darstellung als Abbildung:

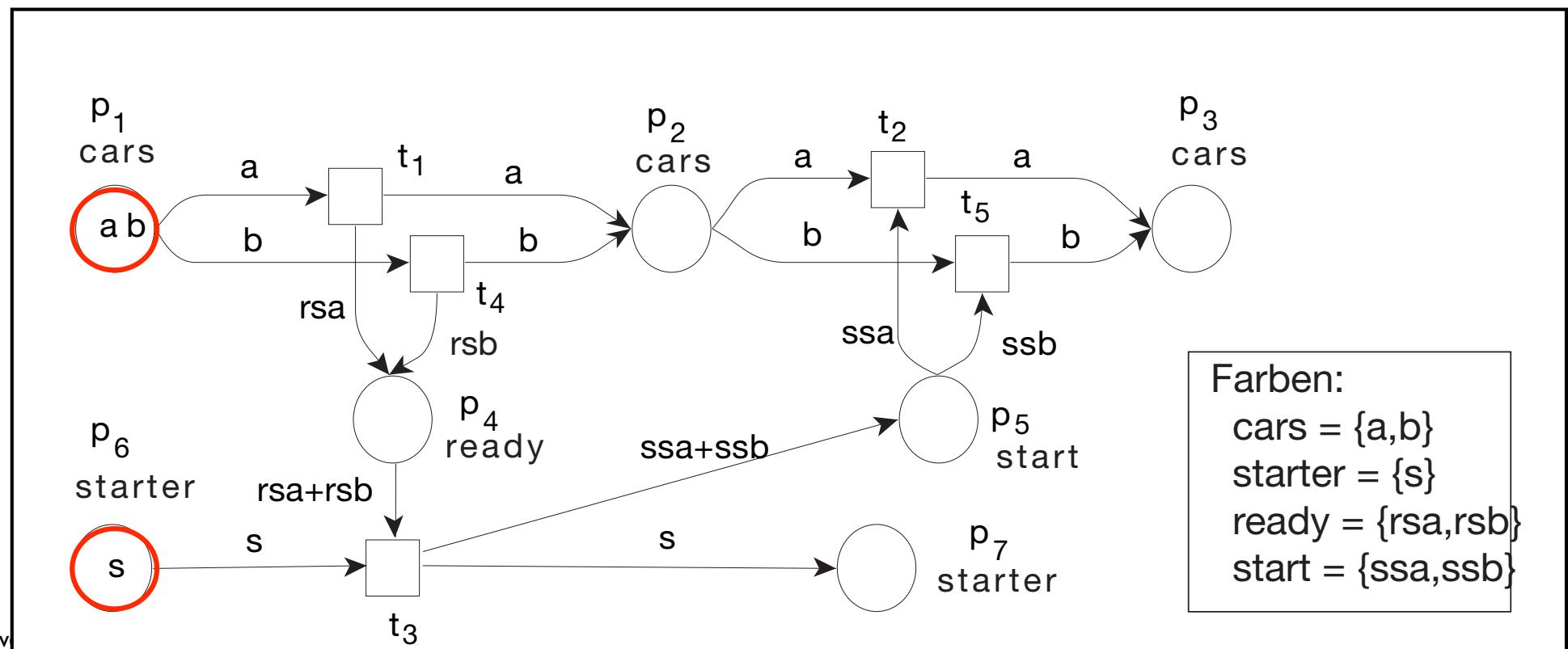
$$\mathbf{m}_0(p_1) = \{a, b\}_b$$

$$\mathbf{m}_0(p_6) = \{s\}_b$$

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

Darstellung als Vektor:

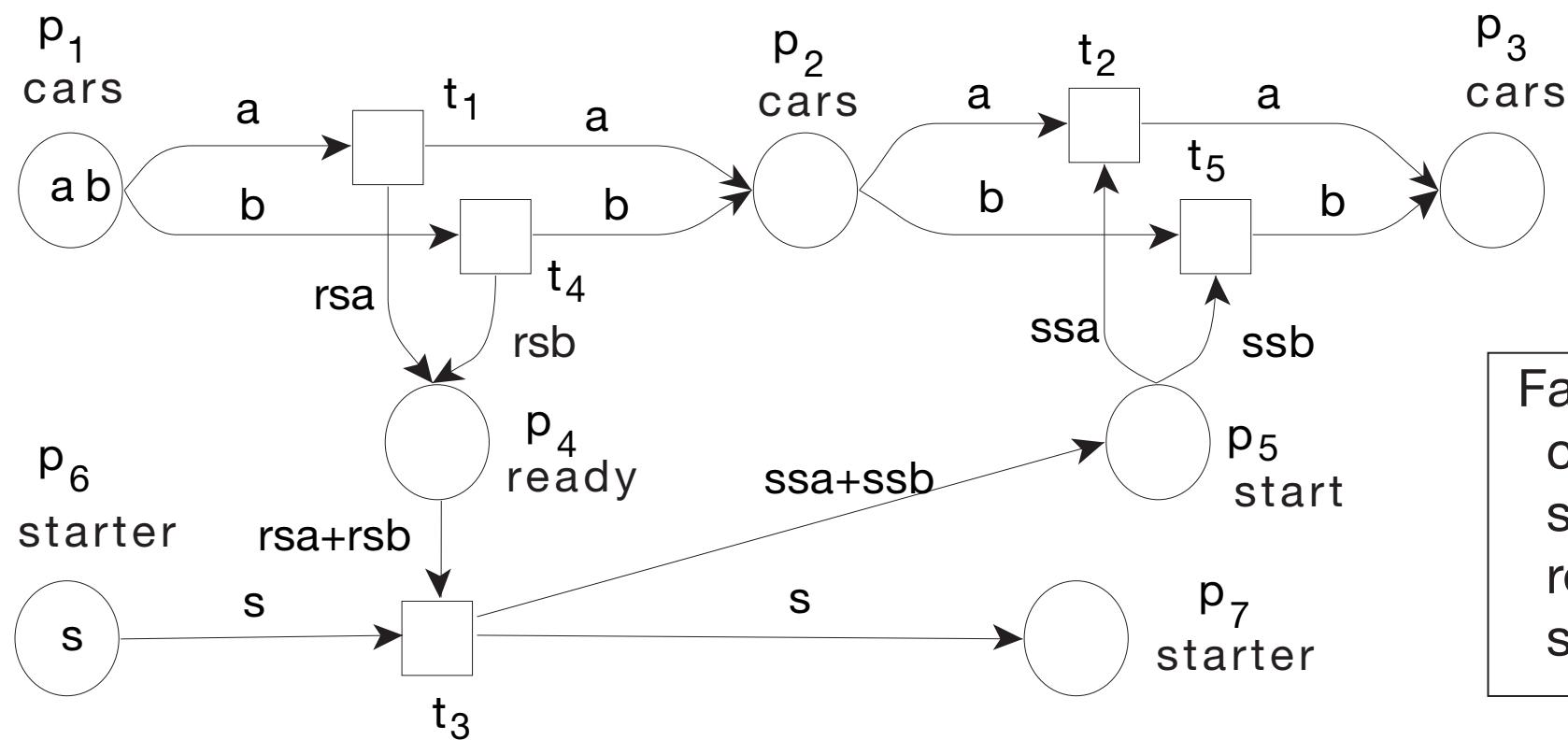
$$p_1 \begin{pmatrix} \{a, b\} \\ \emptyset \\ \emptyset \\ \emptyset \\ \emptyset \\ \emptyset \\ \{s\} \\ \emptyset \end{pmatrix} \xrightarrow{t_4} \begin{pmatrix} \{a\} \\ \{b\} \\ \emptyset \\ \{rsb\} \\ \emptyset \\ \{s\} \\ \emptyset \end{pmatrix} \xrightarrow{t_1} \begin{pmatrix} \emptyset \\ \{a, b\} \\ \emptyset \\ \{rsa, rsb\} \\ \emptyset \\ \{s\} \\ \emptyset \end{pmatrix} \xrightarrow{t_3} \begin{pmatrix} \emptyset \\ \{a, b\} \\ \emptyset \\ \emptyset \\ \{ssa, ssb\} \\ \emptyset \\ \{s\} \end{pmatrix} \xrightarrow{t_2} \begin{pmatrix} \emptyset \\ \{b\} \\ \{a\} \\ \emptyset \\ \{ssb\} \\ \emptyset \\ \{s\} \end{pmatrix} \xrightarrow{t_5} \begin{pmatrix} \emptyset \\ \emptyset \\ \{a, b\} \\ \emptyset \\ \emptyset \\ \emptyset \\ \{s\} \end{pmatrix}$$



Definition 8.3 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: $\mathbf{m} \xrightarrow{t}$).

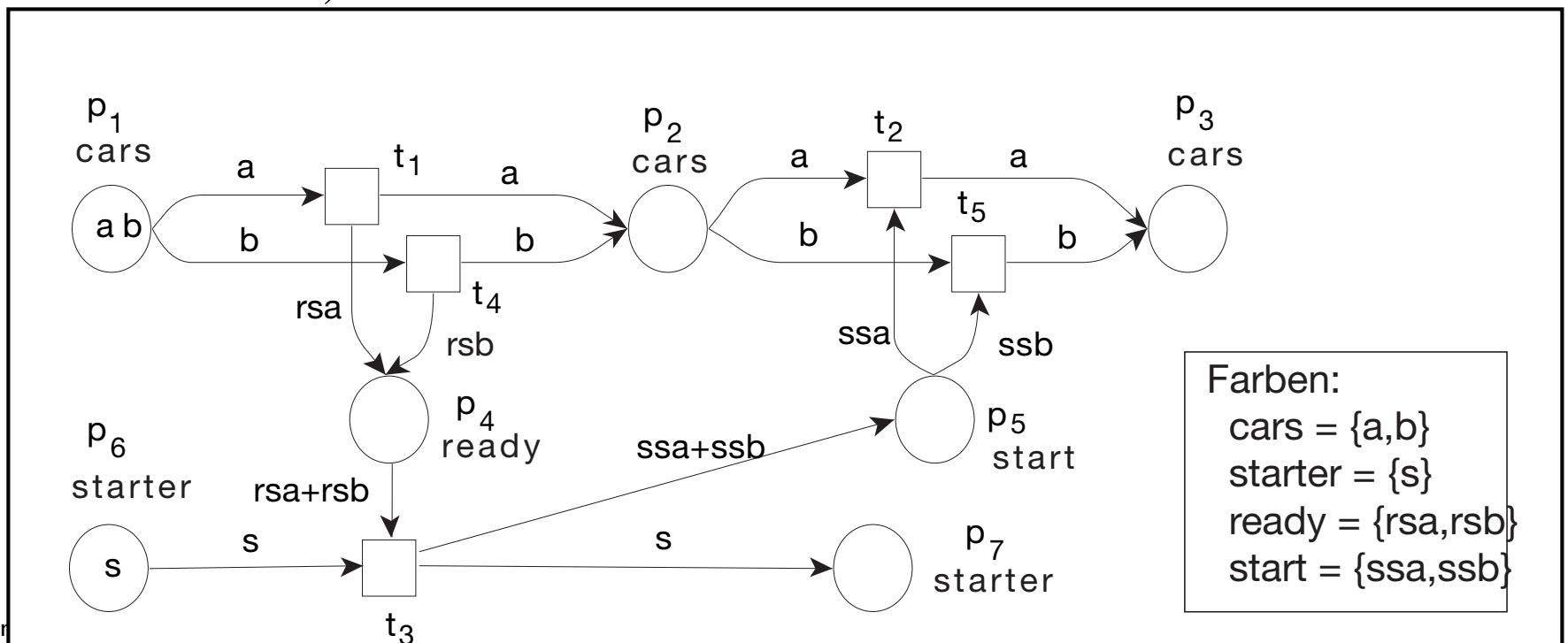


Farben:
cars = {a,b}
starter = {s}
ready = {rsa,rsb}
start = {ssa,ssb}

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

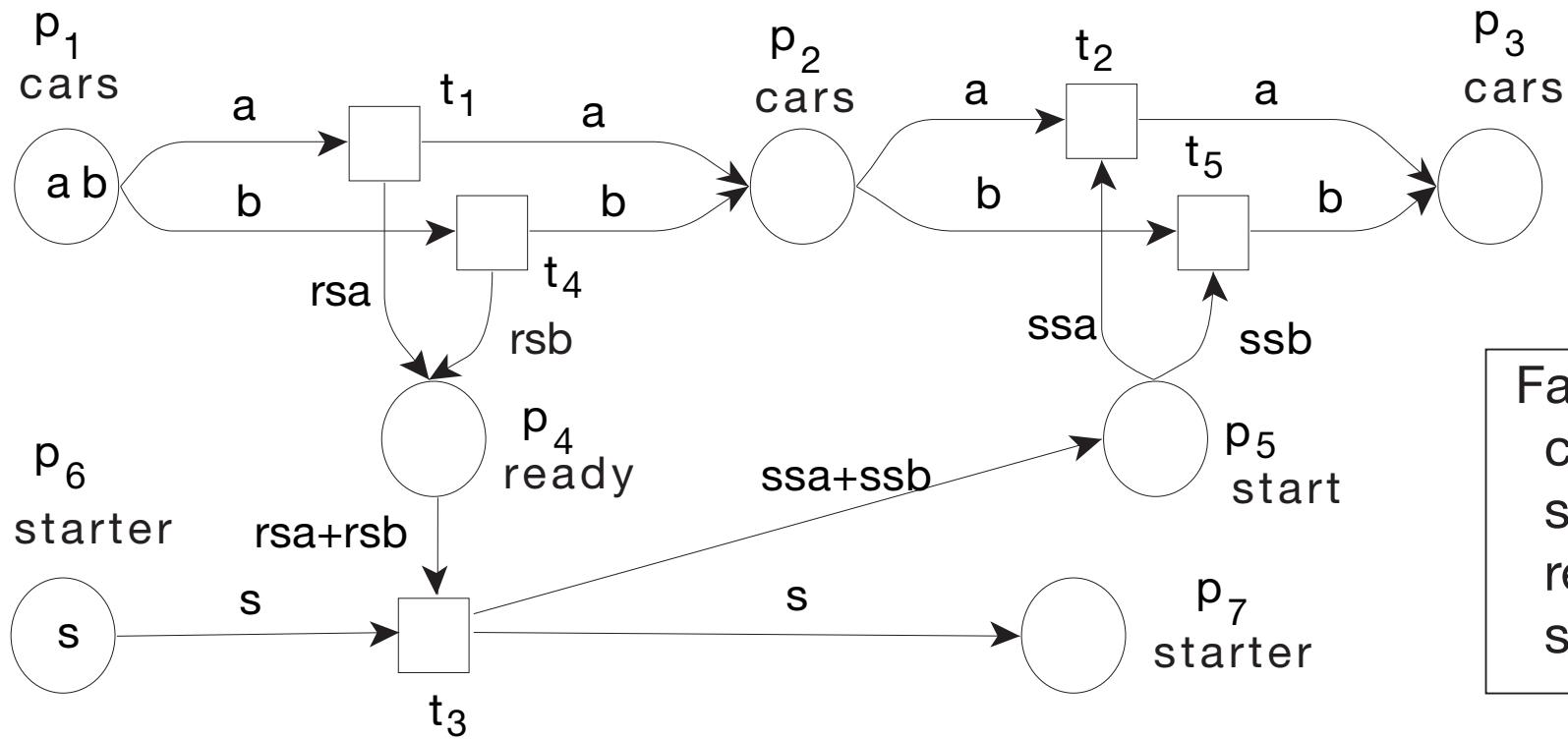
und entsprechend $\widetilde{W}(t, p) := \begin{cases} W(t, p) & \text{falls } (t, p) \in F, \\ \emptyset & \text{sonst.} \end{cases}$

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.

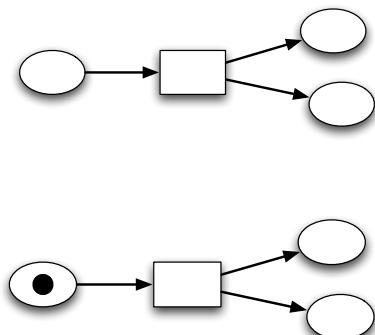


Farben:

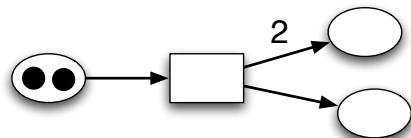
$\text{cars} = \{\text{a,b}\}$
$\text{starter} = \{\text{s}\}$
$\text{ready} = \{\text{rsa,rsb}\}$
$\text{start} = \{\text{ssa,ssb}\}$

Netzklassen

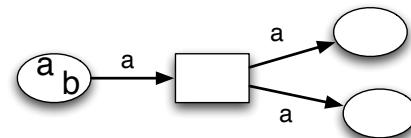
„einfache“ Netze



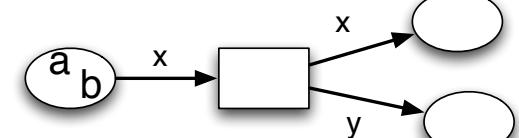
P/T-Netze



kantenkonstante Netze



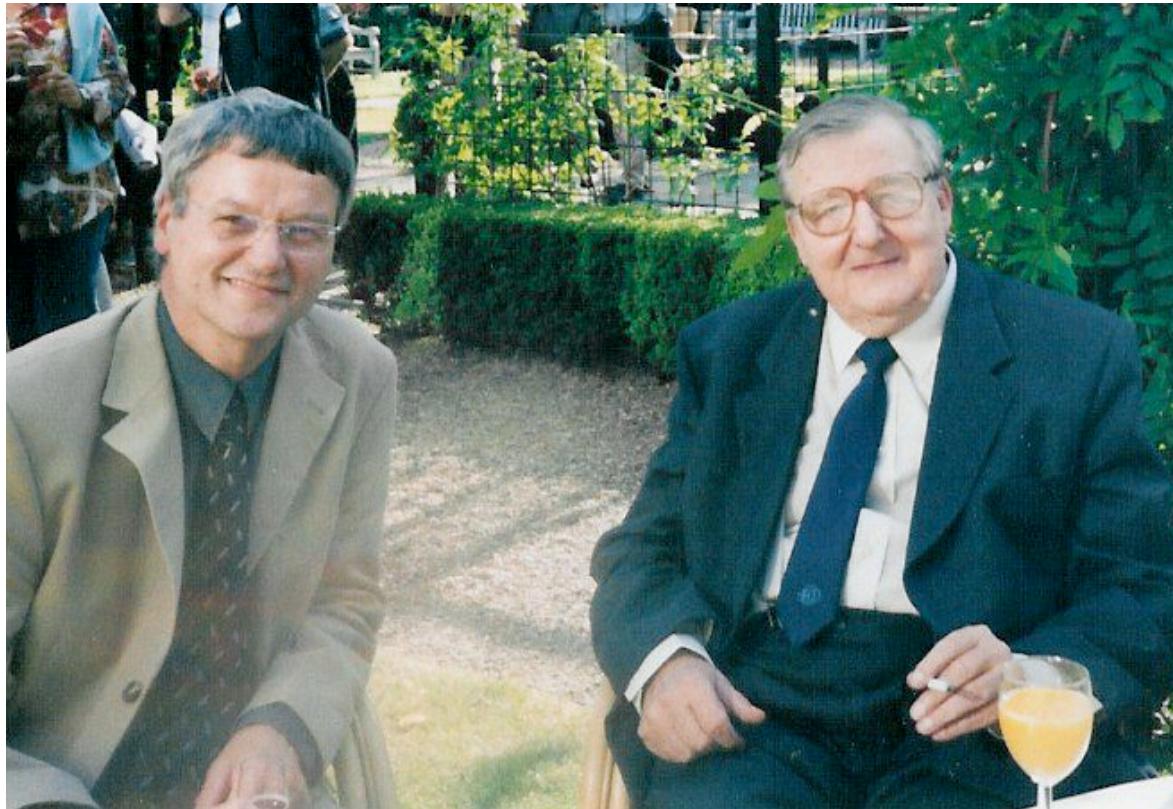
gefärbte Netze



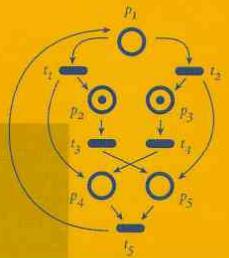
FGI 2

Daniel Moldt

Elementare Systemeigenschaften von P/T-Netzen



Claude Girault
Rüdiger Valk



Petri Nets for Systems Engineering

A Guide to Modeling, Verification,
and Applications

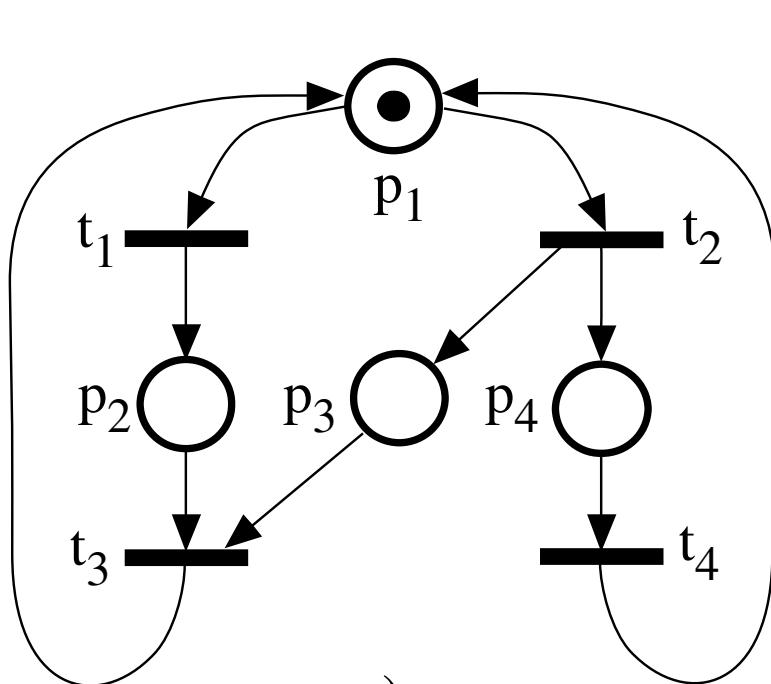


Springer

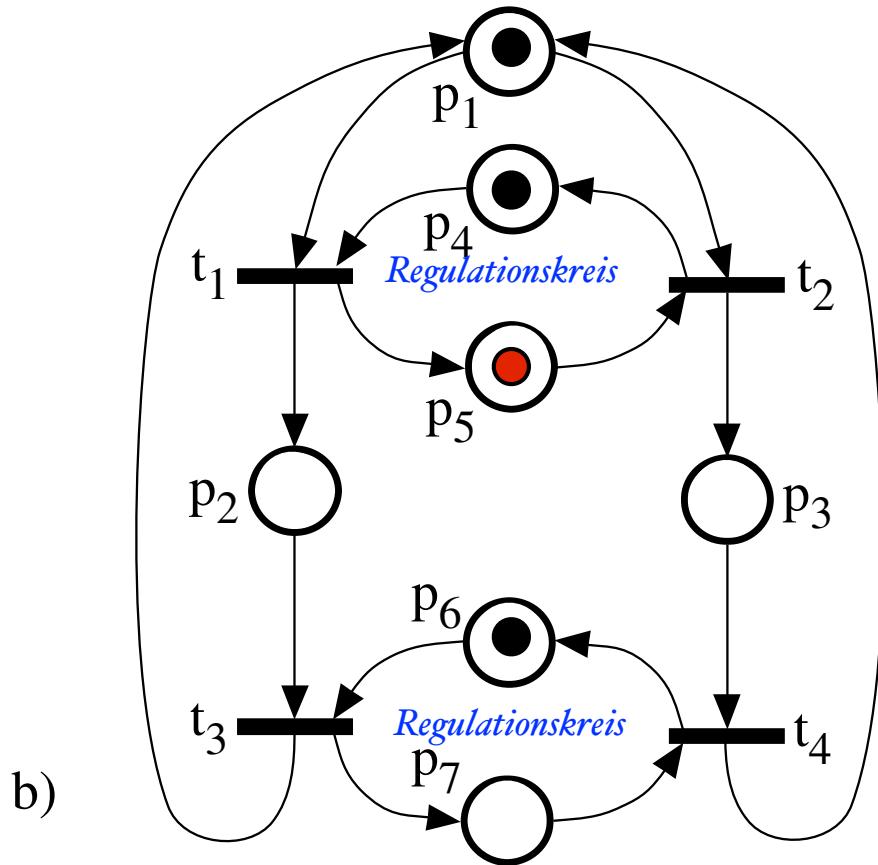
Elementare Systemeigenschaften

- 1) *Beschränktheit (boundedness)*, was die Endlichkeit des Zustandraumes bedeutet,
- 2) *Lebendigkeit (liveness)*, was die potenzielle Ausführbarkeit bedeutet,
- 3) *Reversibilität (reversibility)*, was diejenigen Systeme charakterisiert, die immer in den Anfangszustand zurückgesetzt werden können,
- 4) *wechselseitiger Ausschluss (mutual exclusion)*, was die Unmöglichkeit von simultanen Teilmarkierungen (p-mutex) oder Transitionsausführungen (t-mutex) bedeutet.

Beispiel



a)



b)

- (a) ein unbeschränktes, nicht-lebendiges und nicht-reversibles Netz;
- (b) ein lebendiges Netz, das aber bei vergrößerter Anfangsmarkierung (z.B. $\mathbf{m}_0(p_5) = 1$) nicht lebendig ist.

Beispiel

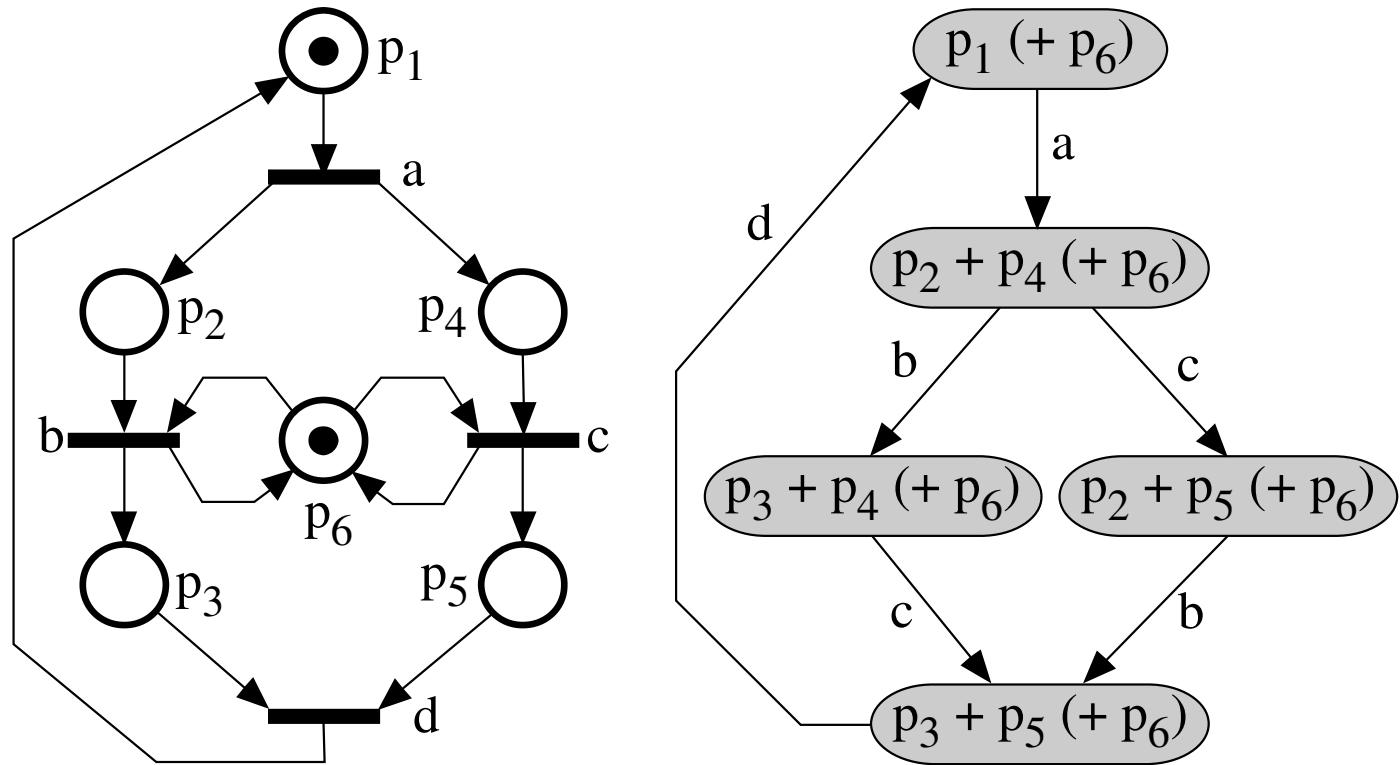
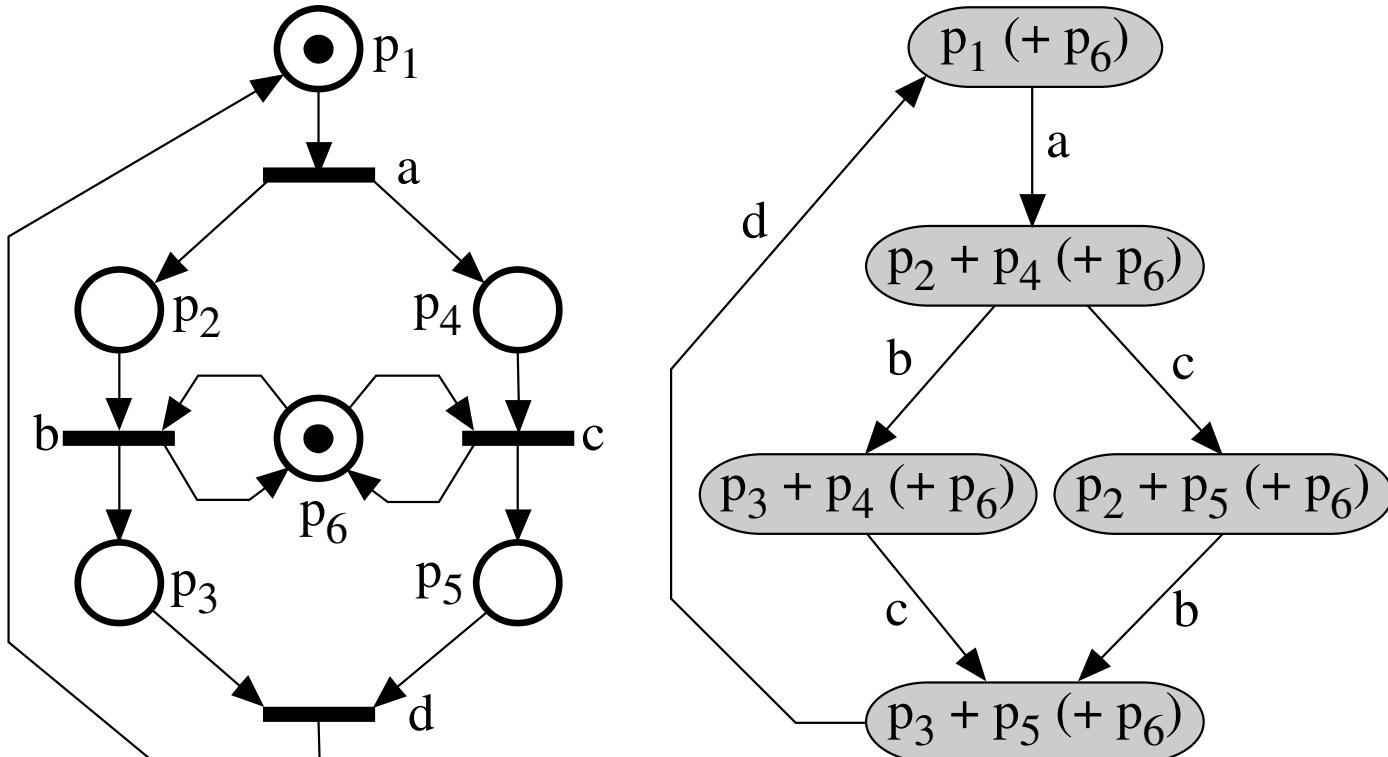


Abbildung 5.2: Beschränktes, lebendiges und reversibles Netz mit Erreichbarkeitsgraph

Platz-Invarianten-Gleichungen



Platz-Invarianten-Gleichungen

$$\mathbf{m}[p_1] + \mathbf{m}[p_2] + \mathbf{m}[p_3] = \mathbf{m}_0[p_1] + \mathbf{m}_0[p_2] + \mathbf{m}_0[p_3] = k_1(\mathbf{m}_0)$$

$$\mathbf{m}[p_1] + \mathbf{m}[p_4] + \mathbf{m}[p_5] = \mathbf{m}_0[p_1] + \mathbf{m}_0[p_4] + \mathbf{m}_0[p_5] = k_2(\mathbf{m}_0)$$

$$\mathbf{m}[p_6] = \mathbf{m}_0[p_6] = k_3(\mathbf{m}_0)$$

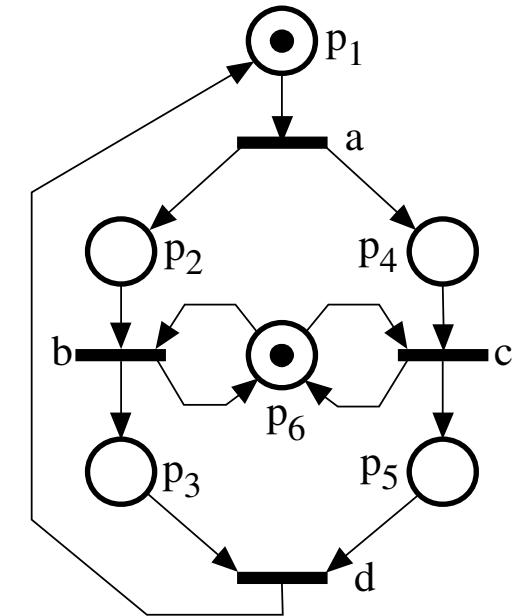
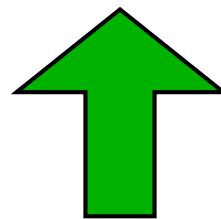
Eigenschaften von erreichbaren Markierungen

$$\mathbf{m}[p_1] \leq \min(k_1(\mathbf{m}_0), k_2(\mathbf{m}_0))$$

$$\mathbf{m}[p_i] \leq k_1(\mathbf{m}_0); i = 2, 3$$

$$\mathbf{m}[p_j] \leq k_2(\mathbf{m}_0); j = 4, 5$$

$$\mathbf{m}[p_6] = k_3(\mathbf{m}_0)$$



Platz-Invarianten-Gleichungen

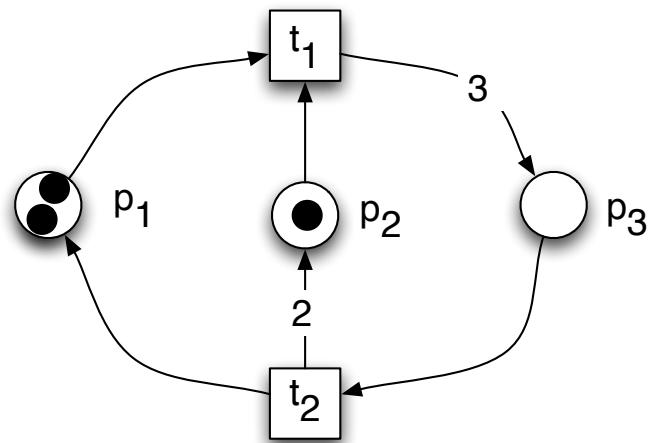
$$\mathbf{m}[p_1] + \mathbf{m}[p_2] + \mathbf{m}[p_3] = \mathbf{m}_0[p_1] + \mathbf{m}_0[p_2] + \mathbf{m}_0[p_3] = k_1(\mathbf{m}_0)$$

$$\mathbf{m}[p_1] + \mathbf{m}[p_4] + \mathbf{m}[p_5] = \mathbf{m}_0[p_1] + \mathbf{m}_0[p_4] + \mathbf{m}_0[p_5] = k_2(\mathbf{m}_0)$$

$$\mathbf{m}[p_6] = \mathbf{m}_0[p_6] = k_3(\mathbf{m}_0)$$

Beschränktheit

- (1) Sei $k \in \mathbb{N}$. Dann heißt ein Platz $p \in P$ **k -beschränkt** (k -bounded) in \mathcal{N} , falls $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m}(p) \leq k$
- (2) p heißt **beschränkt** (bounded) in \mathcal{N} , falls $\exists k \in \mathbb{N} \forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m}(p) \leq k$
- (3) \mathcal{N} heißt **k -beschränkt** bzw. **beschränkt**, wenn alle Plätze k -beschränkt bzw. beschränkt sind.



das “kleinste” unbeschränkte Netz?



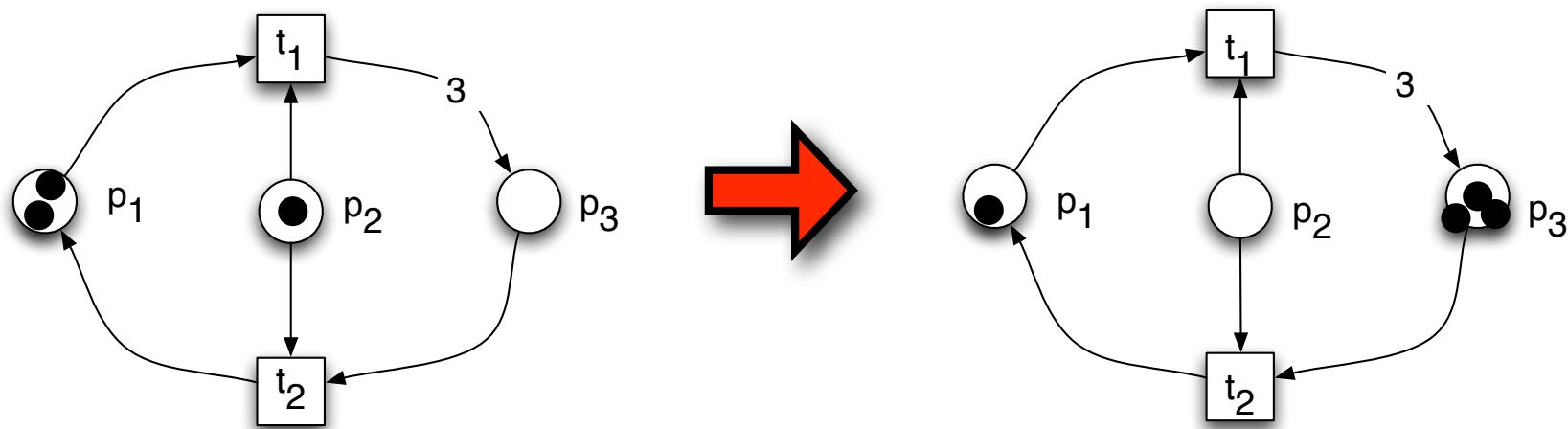
das “kleinste” beschränkte Netz?



$$\forall p \in P \quad \exists k \in \mathbb{N} \quad \forall \mathbf{m} \in \mathbf{R}(\mathcal{N}) : \mathbf{m}(p) \leq k$$

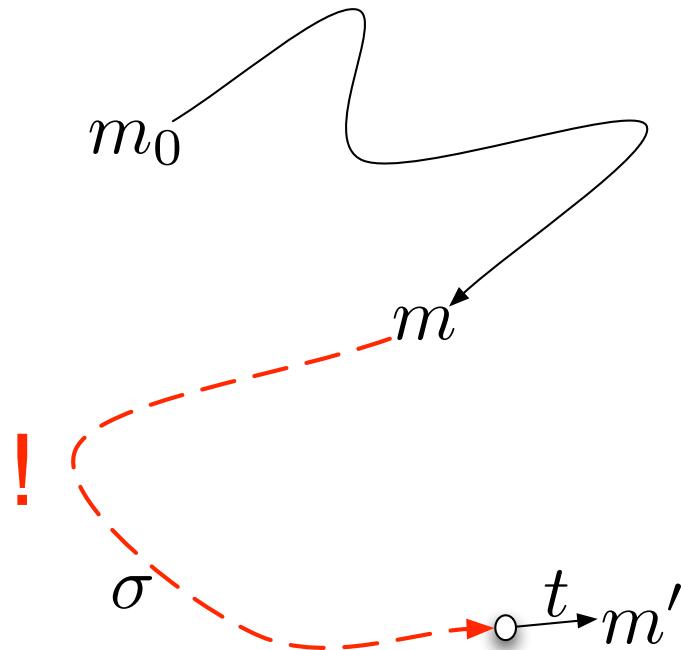
Verklemmungsfreie Netze

(4) $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ heißt *verklemmungsfrei* (deadlock-free) falls
 $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0). \exists t \in T : \mathbf{m} \xrightarrow{t}$



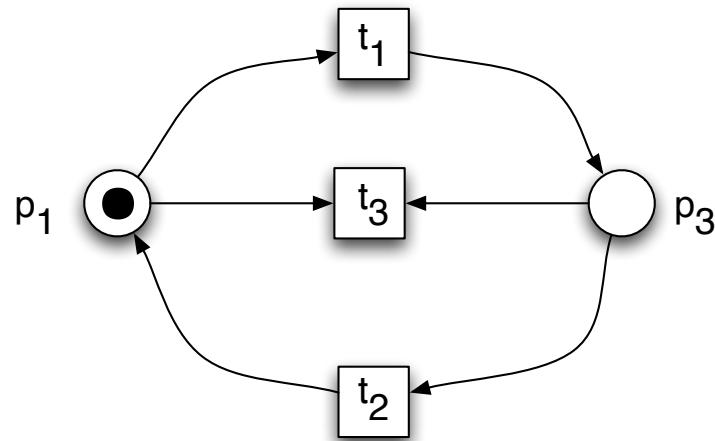
Lebendige Netze

- (4) $\langle \mathcal{N}, m_0 \rangle$ heißt *verklemmungsfrei* (deadlock-free) falls
 $\forall m \in R(\mathcal{N}, m_0). \exists t \in T : m \xrightarrow{t}$
- (5) t heißt *lebendig* (live) in $\langle \mathcal{N}, m_0 \rangle$ falls
 $\forall m \in R(\mathcal{N}, m_0). \exists \sigma \in T^* : m \xrightarrow{\sigma t} m'$
- (6) $\langle \mathcal{N}, m_0 \rangle$ heißt *lebendig* falls alle Transitionen lebendig sind.



Lebendige Netze

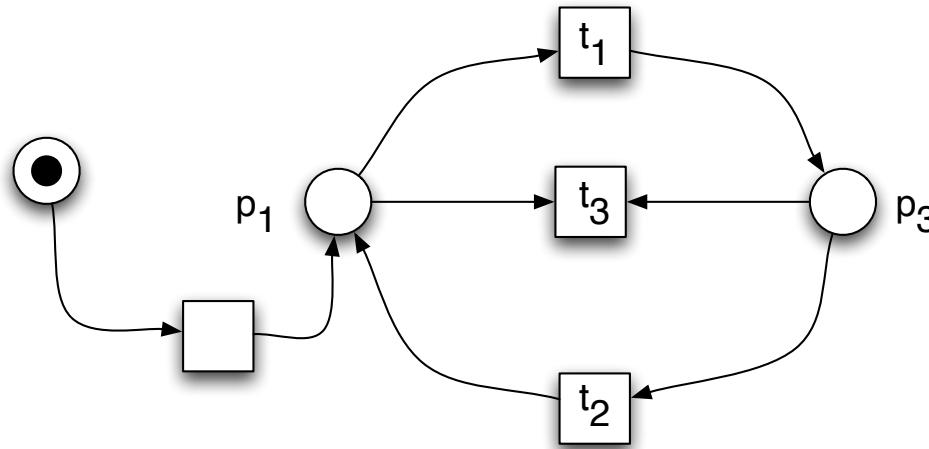
- (4) $\langle \mathcal{N}, m_0 \rangle$ heißt *verklemmungsfrei* (deadlock-free) falls
 $\forall m \in R(\mathcal{N}, m_0). \exists t \in T : m \xrightarrow{t}$
- (5) t heißt *lebendig* (live) in $\langle \mathcal{N}, m_0 \rangle$ falls
 $\forall m \in R(\mathcal{N}, m_0). \exists \sigma \in T^* : m \xrightarrow{\sigma t} m'$
- (6) $\langle \mathcal{N}, m_0 \rangle$ heißt *lebendig* falls alle Transitionen lebendig sind.



verklemmungsfrei, aber
nicht lebendig

Reversibilität, Rücksetzzustände

- (7) $\mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0)$ heißt a *Rücksetzzustand* (home state) falls
 $\forall \mathbf{m}' \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0). \exists \sigma \in T^*: \mathbf{m}' \xrightarrow{\sigma} \mathbf{m}$
- (8) $\langle \mathcal{N}, \mathbf{m}_0 \rangle$ heißt *reversibel* (reversible) falls
 $\forall \mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0). \exists \sigma \in T^*: \mathbf{m} \xrightarrow{\sigma} \mathbf{m}_0$



Ein Netz ist reversibel, falls \mathbf{m}_0 ein Rücksetzzustand ist.

Ausschluss

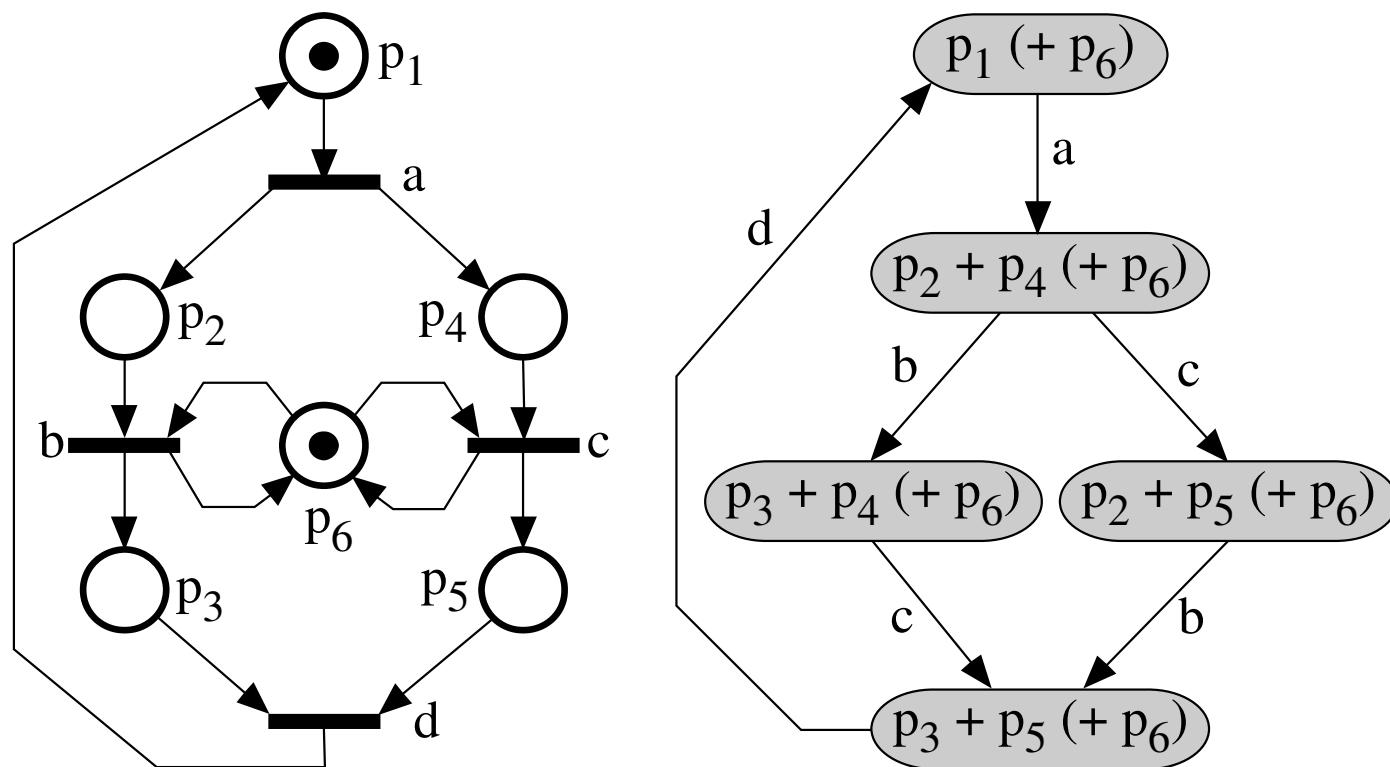
(9) wechselseitiger Ausschluss (mutual exclusion) in $\langle \mathcal{N}, \mathbf{m}_0 \rangle$:

p_i und p_j sind in *Markierungs-Ausschluss* (marking mutual exclusion) falls

$$\nexists \mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0) : (\mathbf{m}[p_i] > 0) \wedge (\mathbf{m}[p_j] > 0)$$

t_i und t_j sind in *Schalt-Ausschluss* (firing mutual exclusion) falls

$$\nexists \mathbf{m} \in \mathbf{R}(\mathcal{N}, \mathbf{m}_0) : \mathbf{m} \geq W(\bullet, t_i) + W(\bullet, t_j)]$$



FGI 2

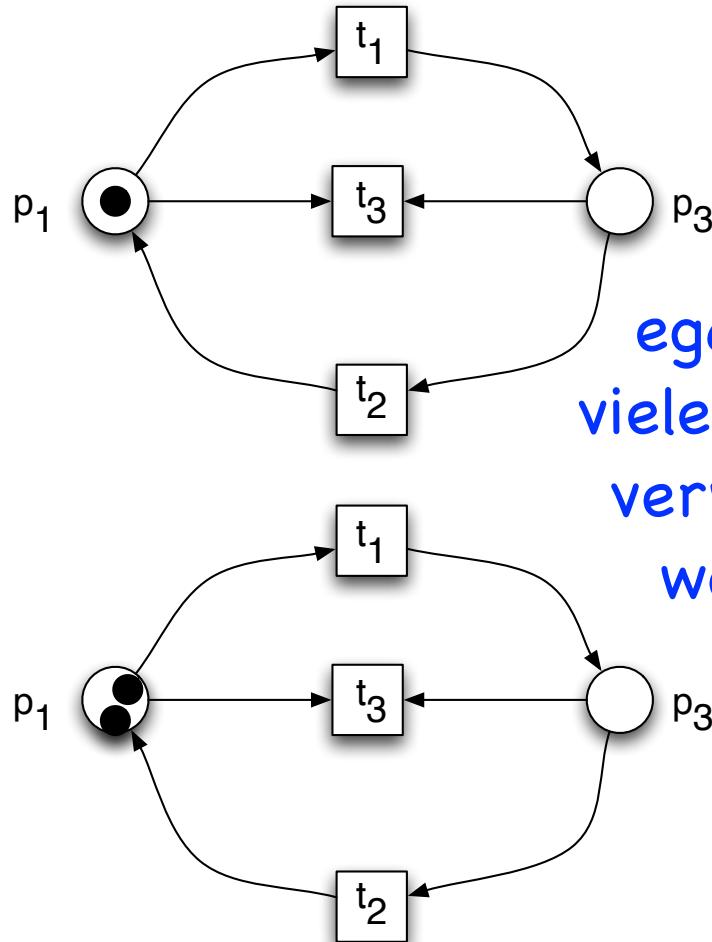
Daniel Moldt

Strukturelle Systemeigenschaften

Strukturelle Beschränktheit

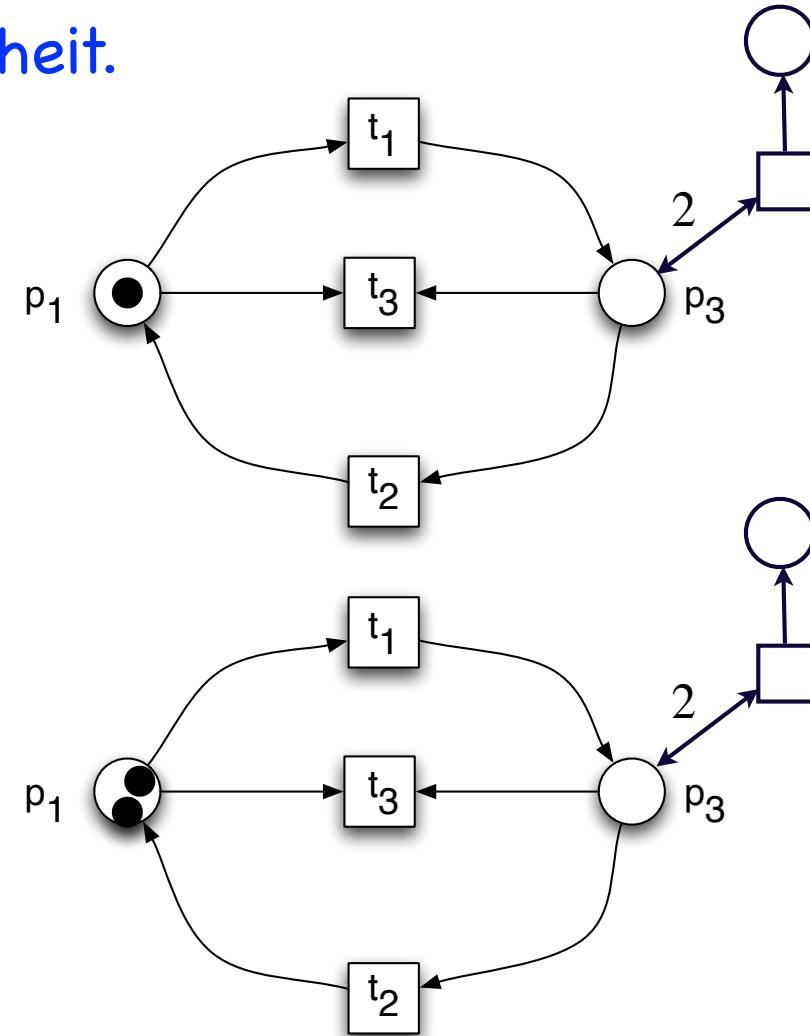
\mathcal{N} heißt *strukturell beschränkt* (structurally bounded), falls $\langle \mathcal{N}, m \rangle$ für alle m beschränkt ist.

Die Struktur erzwingt Beschränktheit.



egal, wie
viele Marken
verwendet
werden

strukturell beschränkt

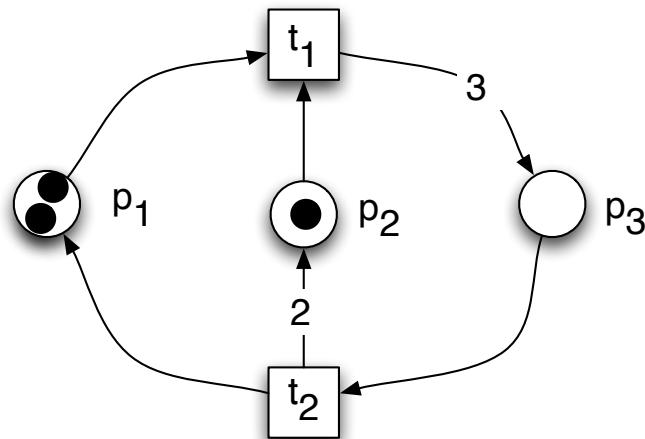


strukturell **nicht** beschränkt

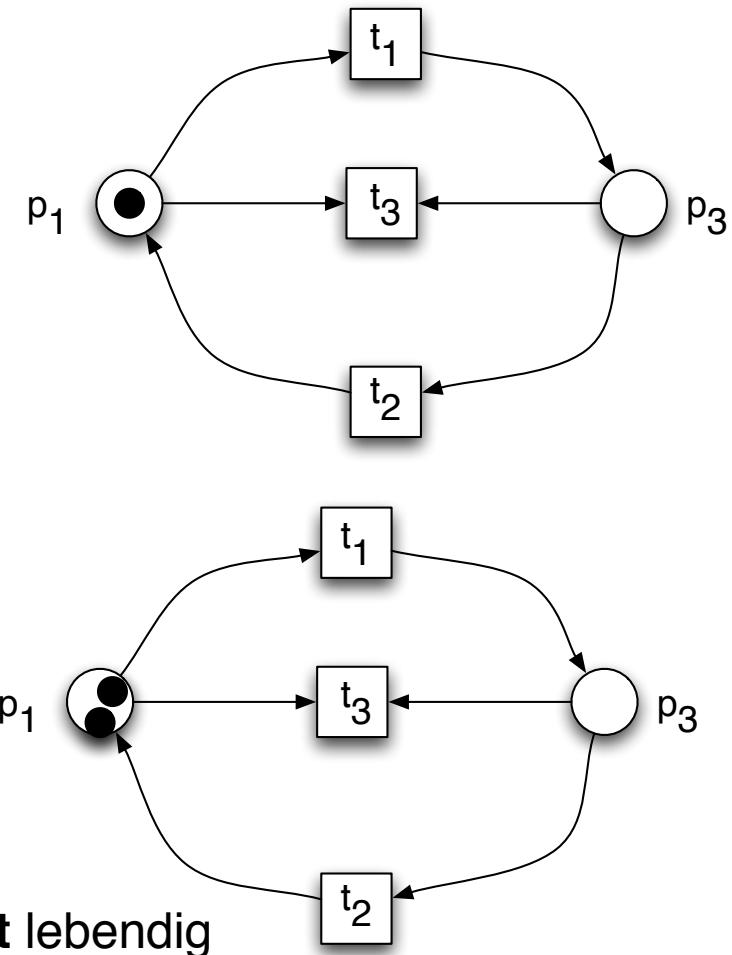
Strukturelle Lebendigkeit

\mathcal{N} heißt *strukturell lebendig* (structurally live), falls $\langle \mathcal{N}, m \rangle$ für mindestens ein m lebendig ist.

Die Struktur erlaubt eine lebendige Anfangsmarkierung.



strukturell lebendig



strukturell **nicht** lebendig

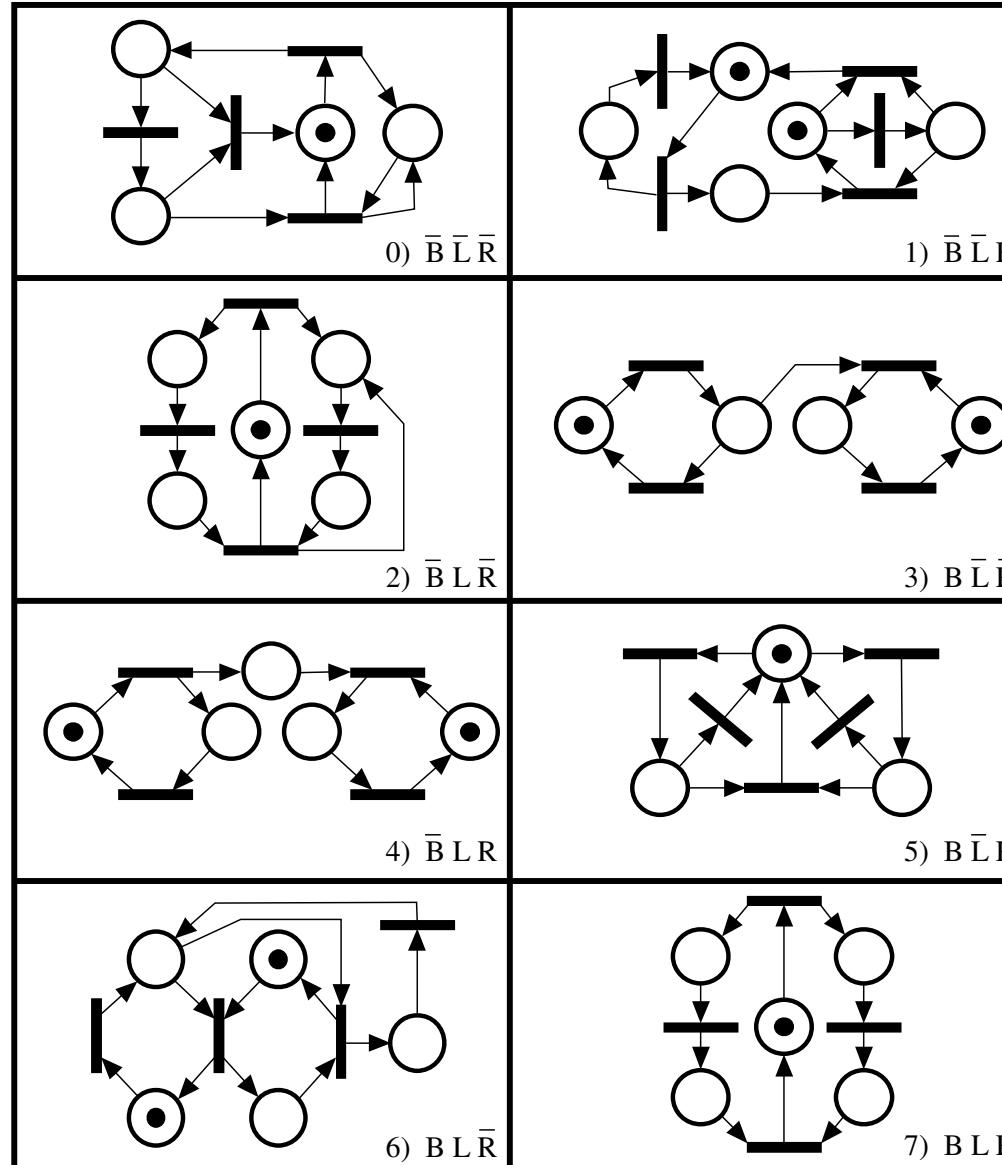
FGI 2

Daniel Moldt

**Unabhängigkeit von
Beschränktheit,
Lebendigkeit und
Reversibilität**

Unabhängigkeit

Lemma: Lebendigkeit, Beschränktheit und Reversibilität sind voneinander unabhängige Systemeigenschaften.



0	1
2	3
4	5
6	7

Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

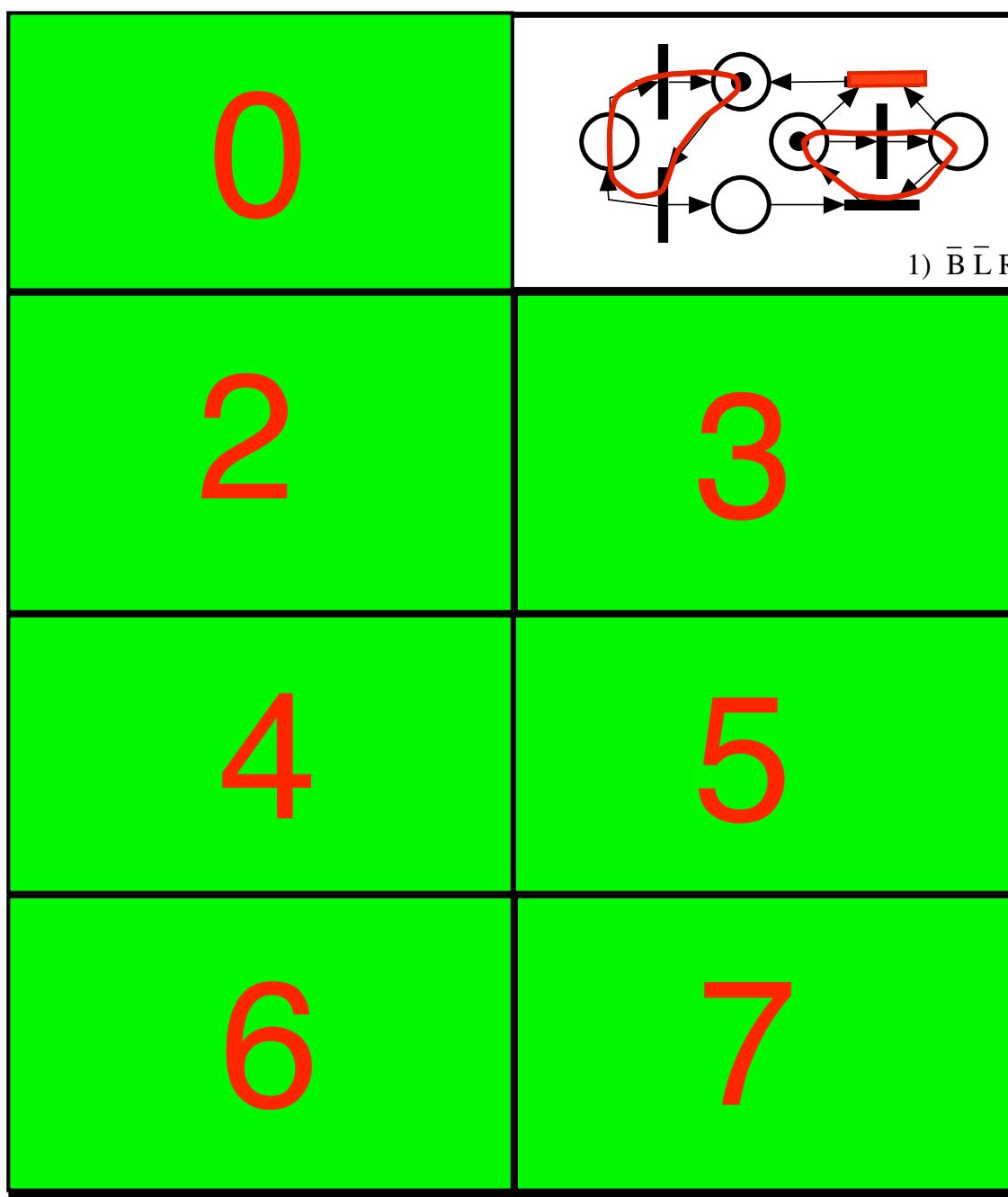


Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

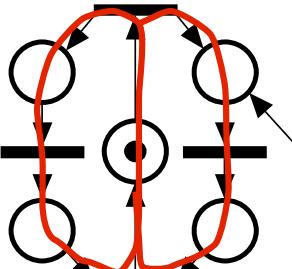
0	1
 2) $\bar{B} \ L \ \bar{R}$	3
4	5
6	7

Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

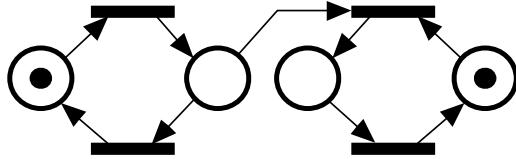
0	1
2	 3) B L R
4	5
6	7

Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

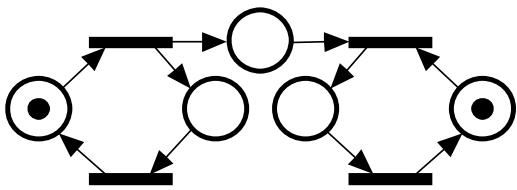
0	1
2	3
 4) $\bar{B} \ L \ R$	5
6	7

Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

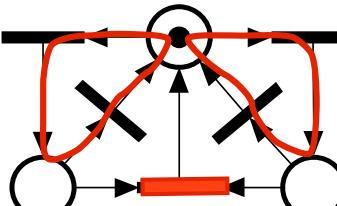
0	1
2	3
4	 5) B L R
6	7

Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

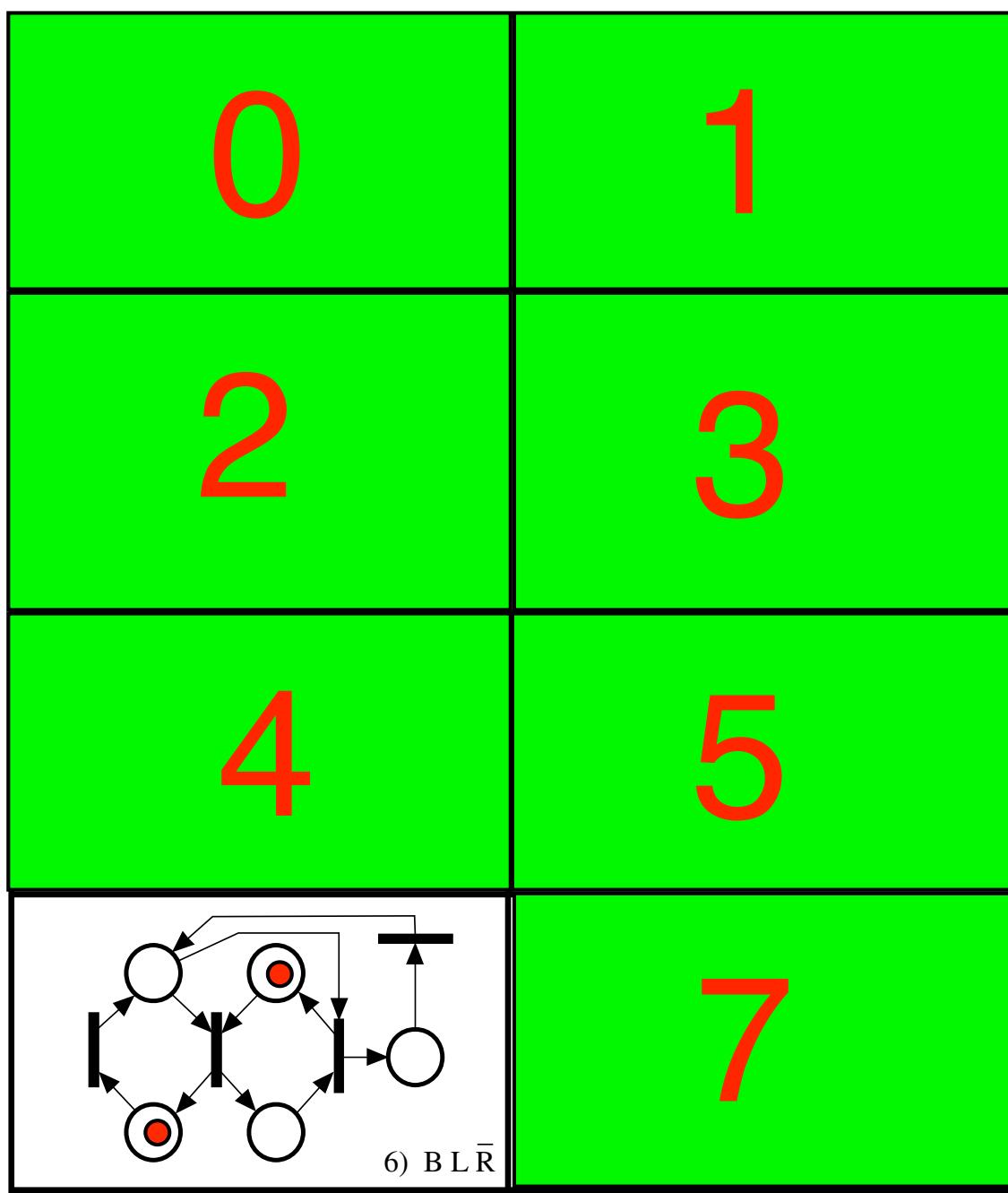


Abbildung 5.4: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

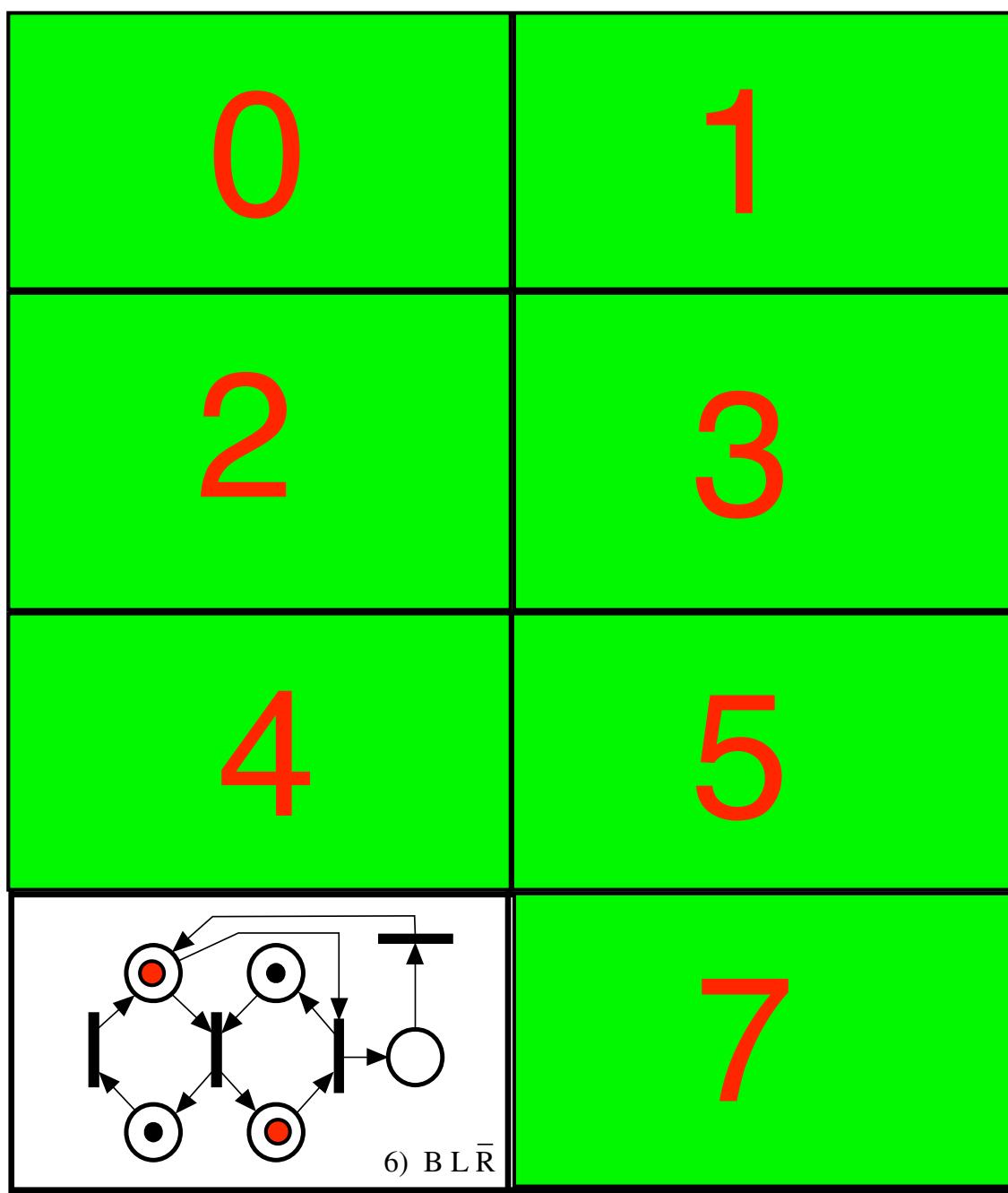


Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

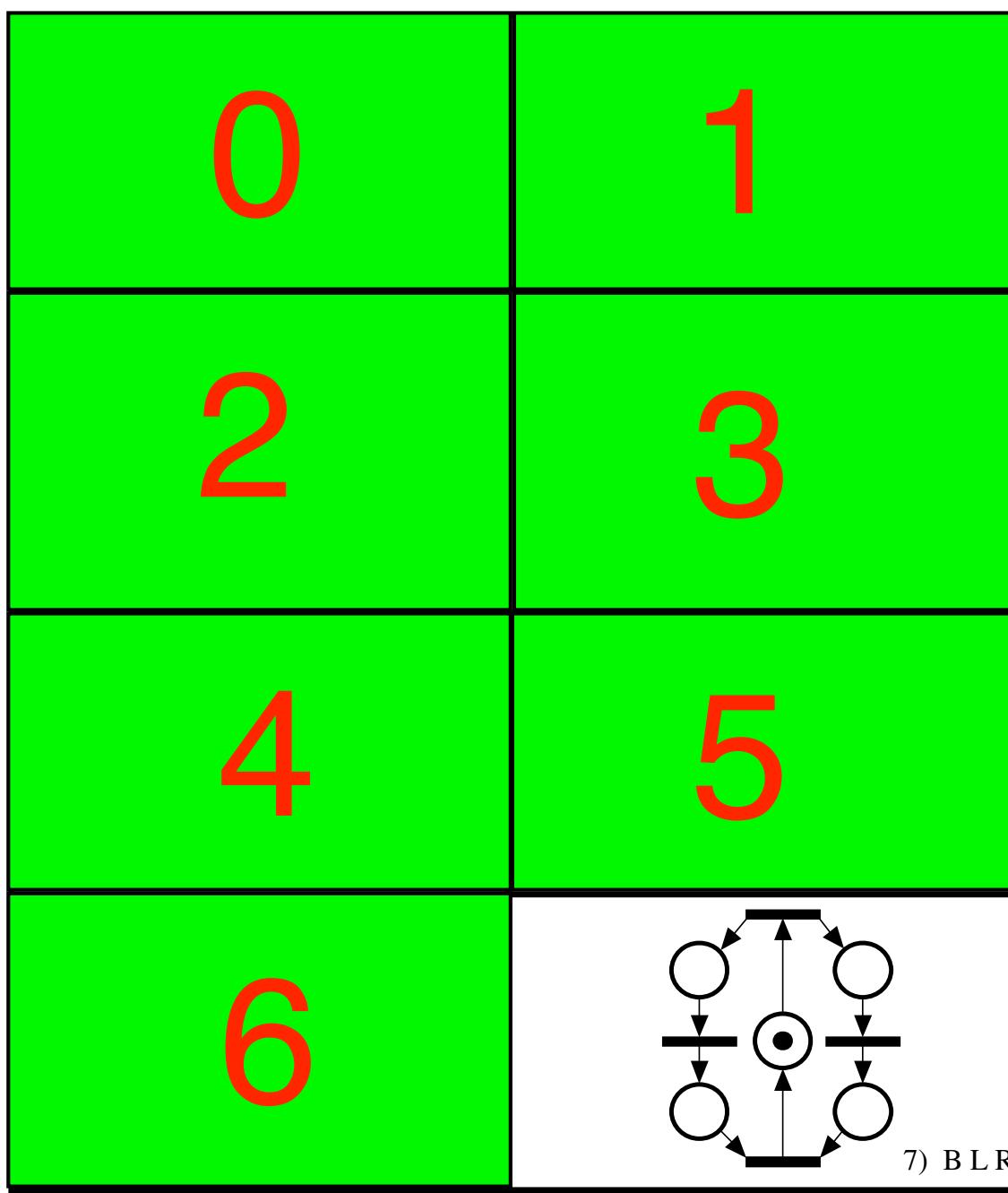


Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften

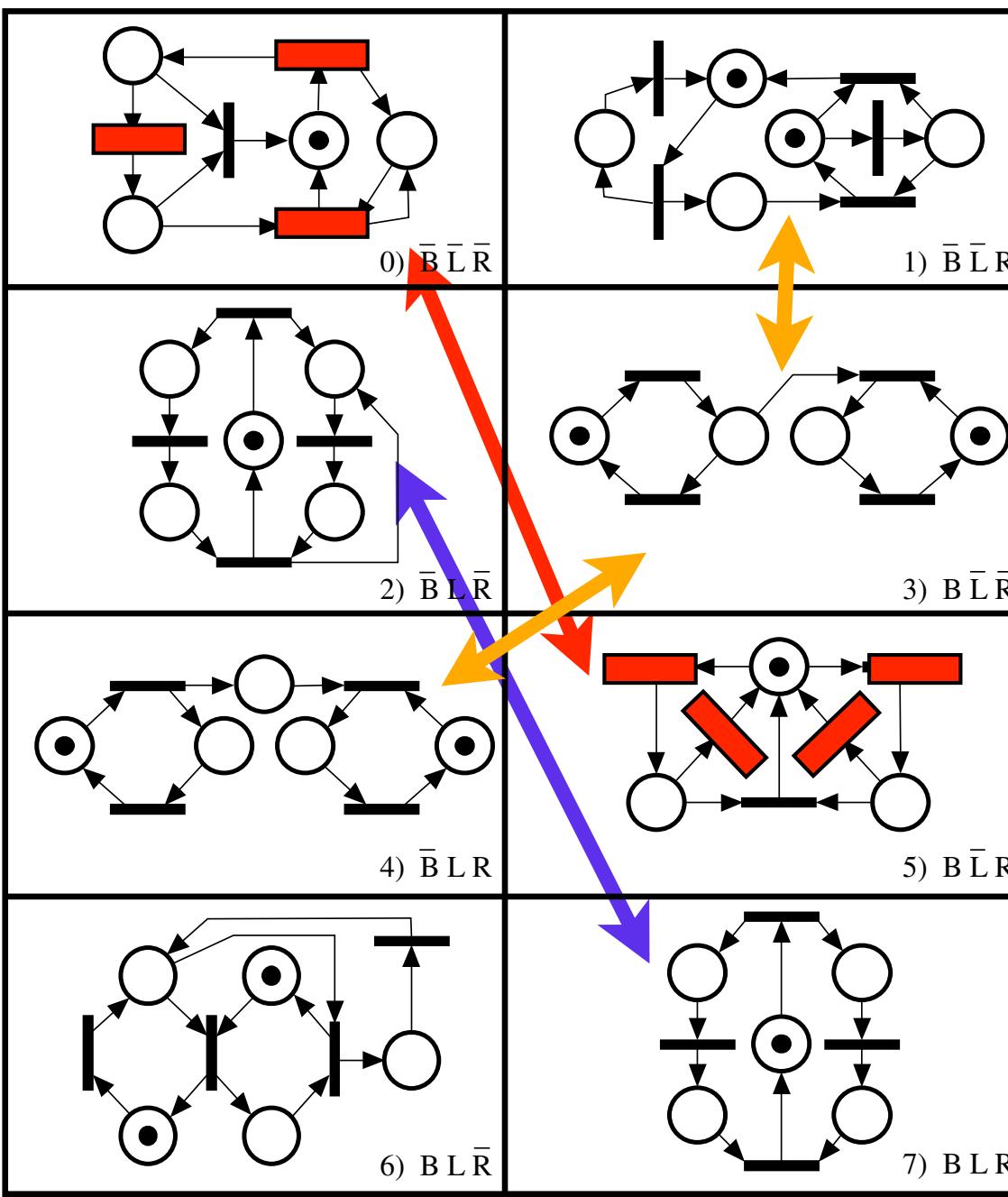


Abbildung 6.19: Beschränktheit (B), Lebendigkeit (L) und Reversibilität (R) sind unabhängige Eigenschaften