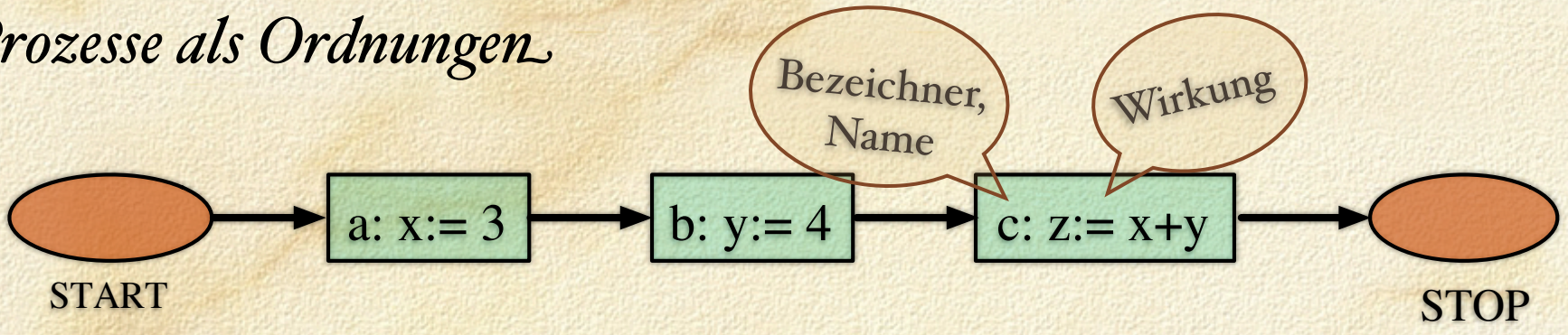
Kapitel 2
Partielle Ordnungen

2 Partielle Ordnungen

2.1 Partielle und strikte Halbordnung . . . . . .

2.2 Logische und vektorielle Zeitstempel . . . . . .

Prozesse als Ordnungen



Eigenschaften der Handlungen:

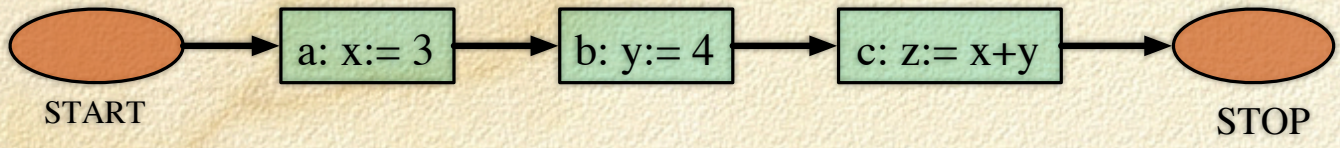
- *extensional*, d.h. durch ihre Wirkung beschreibbar
- *unteilbar* (auch atomar), d.h. sie werden vom Prozessor ununterbrochen ausgeführt

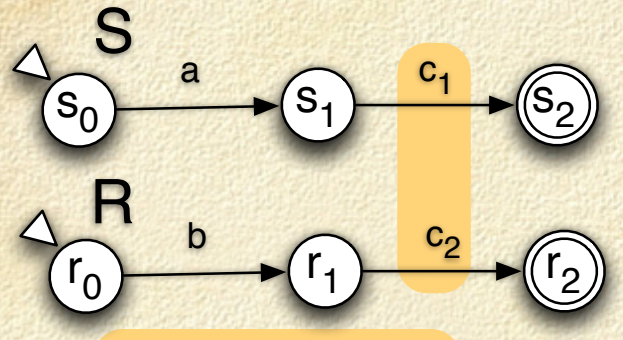
„atomare Operation“

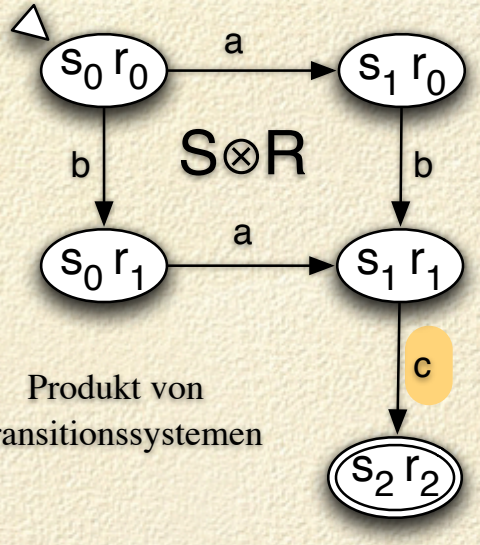
- *geordnet*

a) durch eine totale Ordnung: sequentieller Prozess

Prozesse als Ordnungen





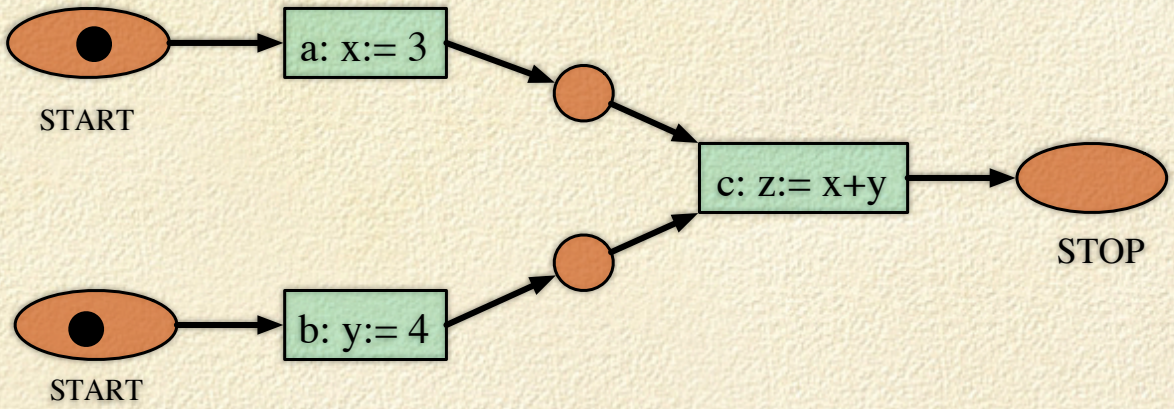


- *geordnet*

a) durch eine totale Ordnung: sequentieller Prozess

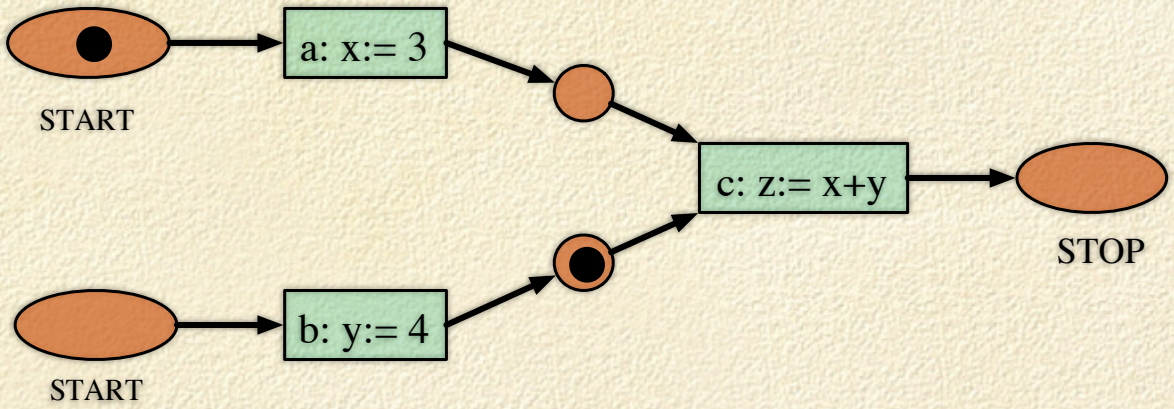
b) partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

als Petrinetz



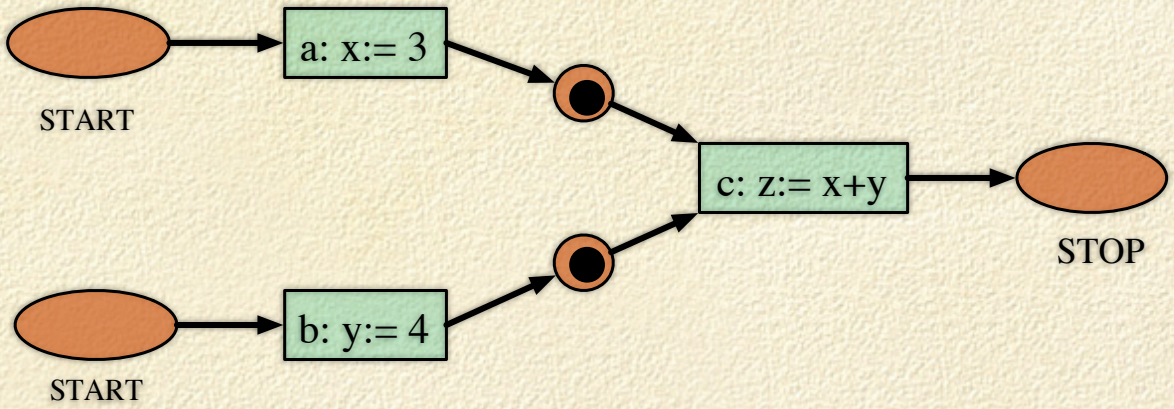
b) partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

als Petrinetz



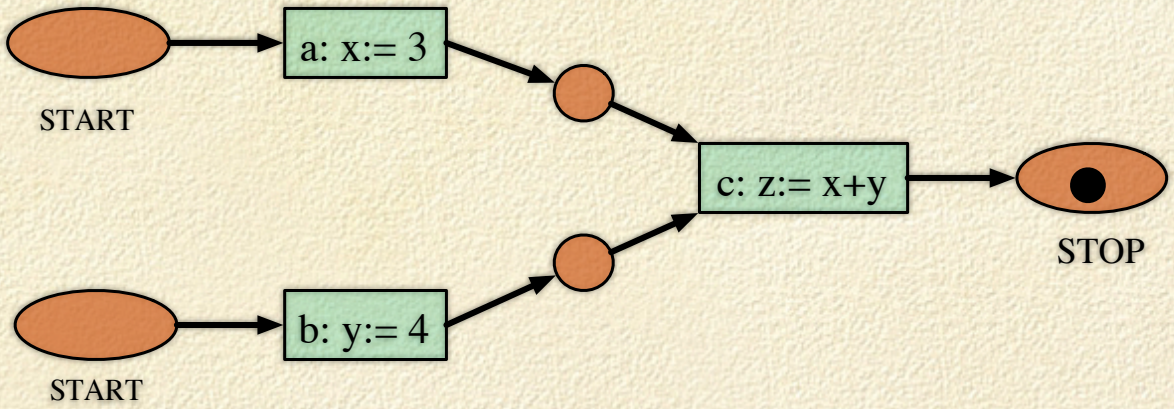
b) partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

als Petrinetz



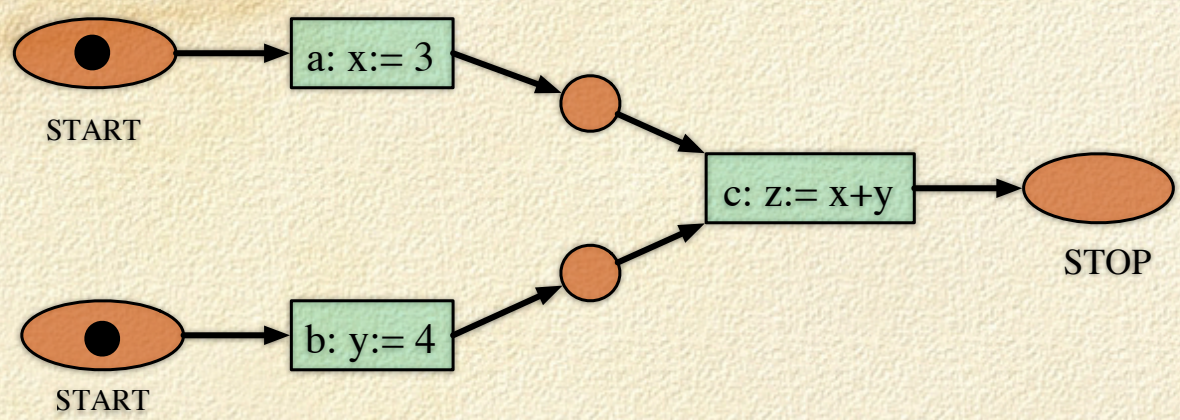
b) partielle Ordnung: nichtsequentieller Prozess, d.h. zeitlich/kausal unabhängige Handlungen sind möglich

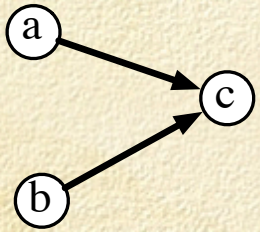
als Petrinetz



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

als Petrinetz





als Striktordnung
oder partielle Ordnung

als (Aktions-)Folgen:

a;b;c und b;a;c

partial order semantics
PO-Semantik

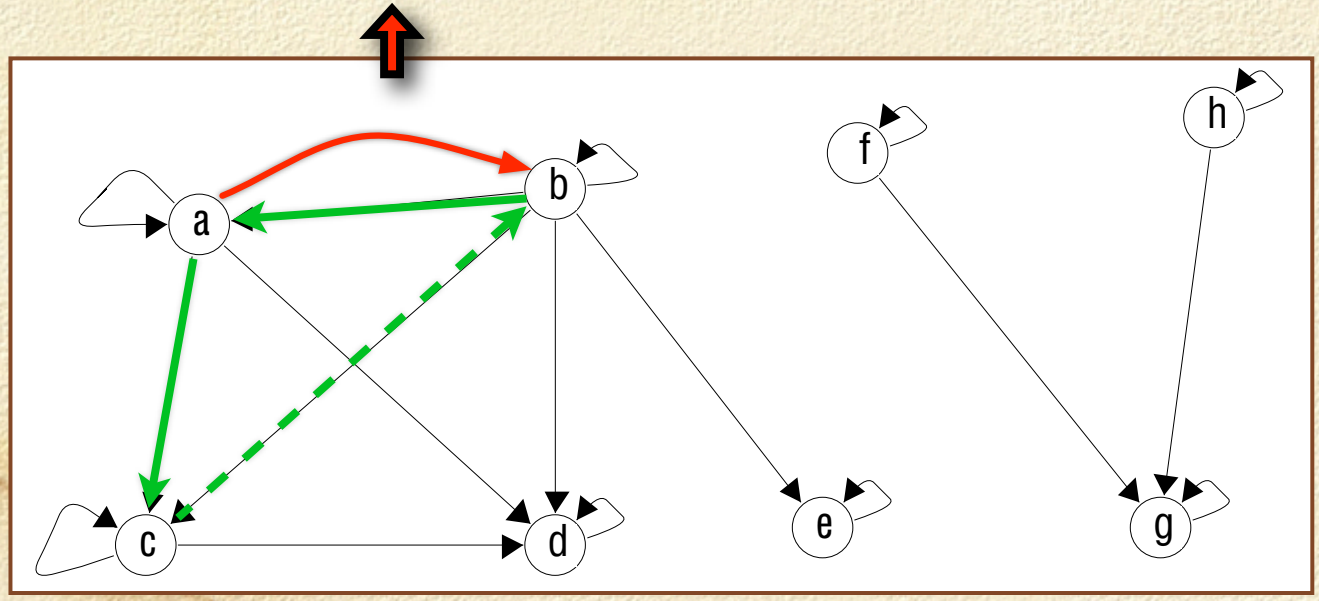
interleaving semantics
Folgen-Semantik

Definition 2.2 *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. $\forall a \in A.\ (a, a) \in R$ "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: $a \leq b$ für $(a, b) \in R$



Zyklen?

keine Zyklen !!!

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

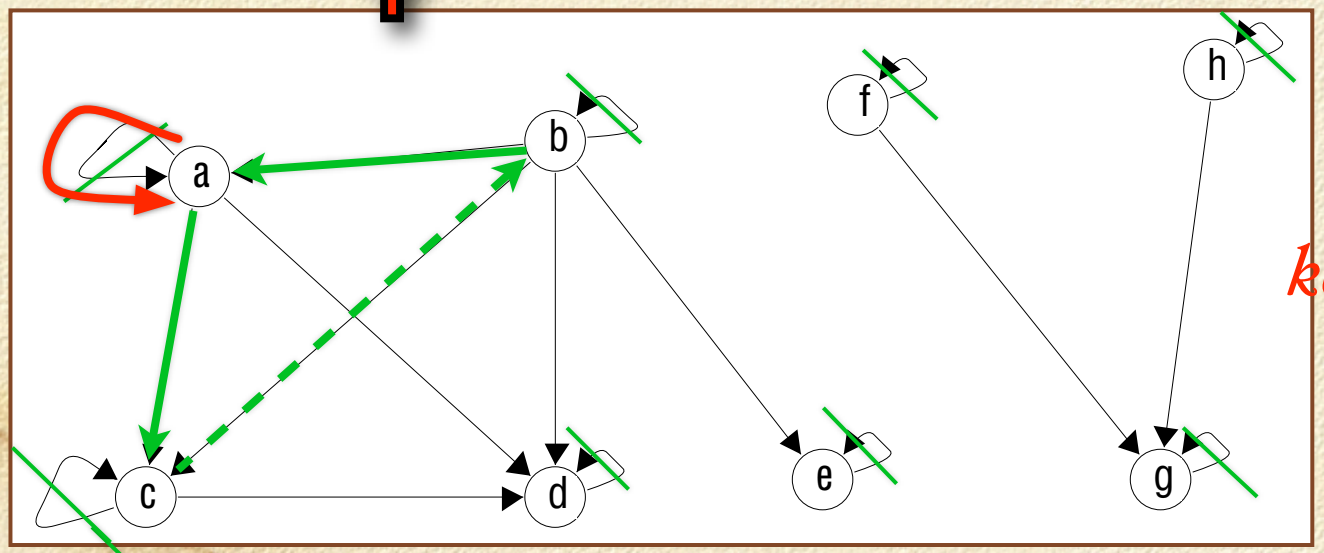
Definition 2.2 *Sei A eine Menge und $R \subseteq A \times A$ eine (binäre) Relation.*

b) *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"

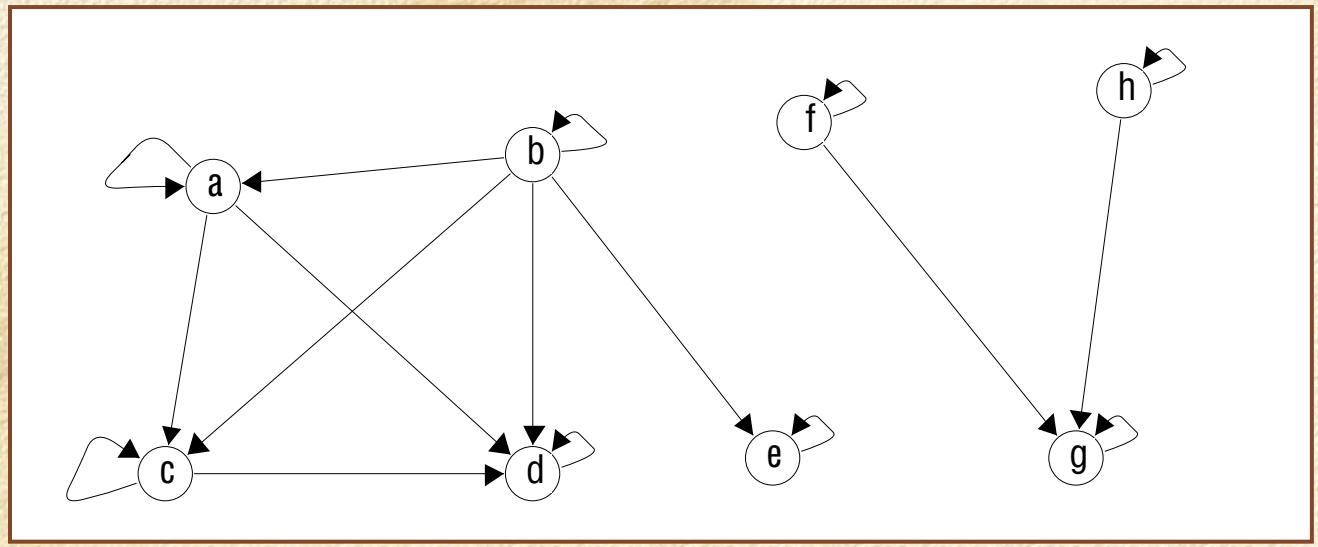
Schreibweise: $a < b$ für $(a, b) \in R$



Zyklen?

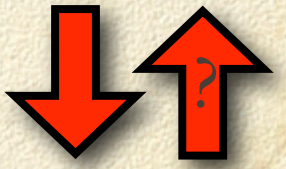
keine Zyklen !!!

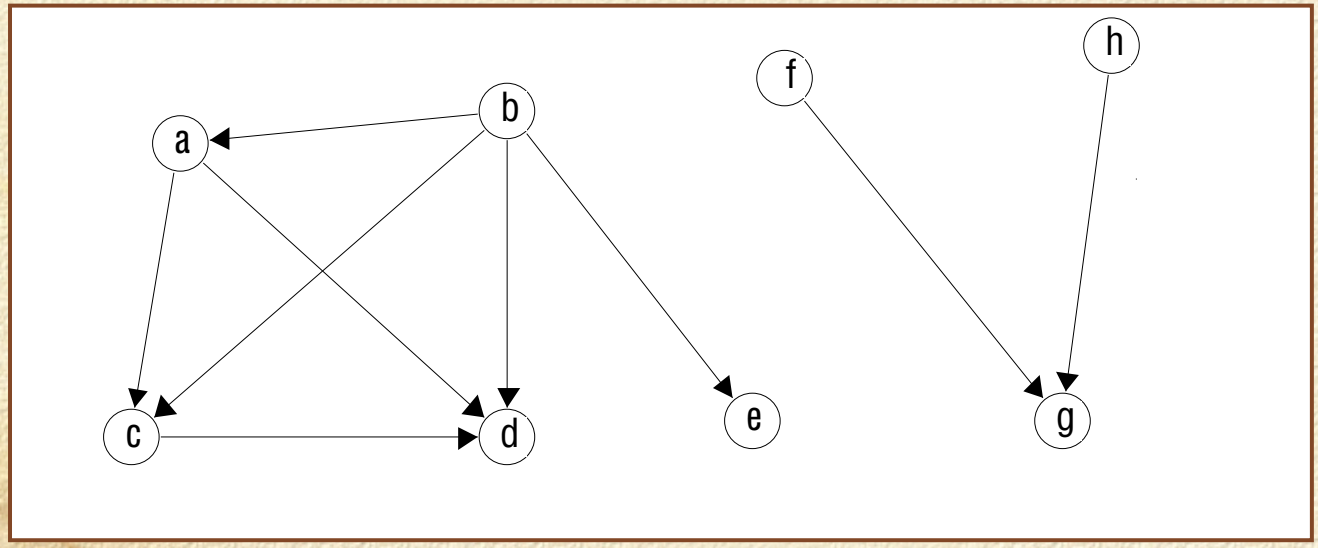
$$R = \{(a,a),(a,c),(b,a),\ldots\}$$



partielle Ordnung

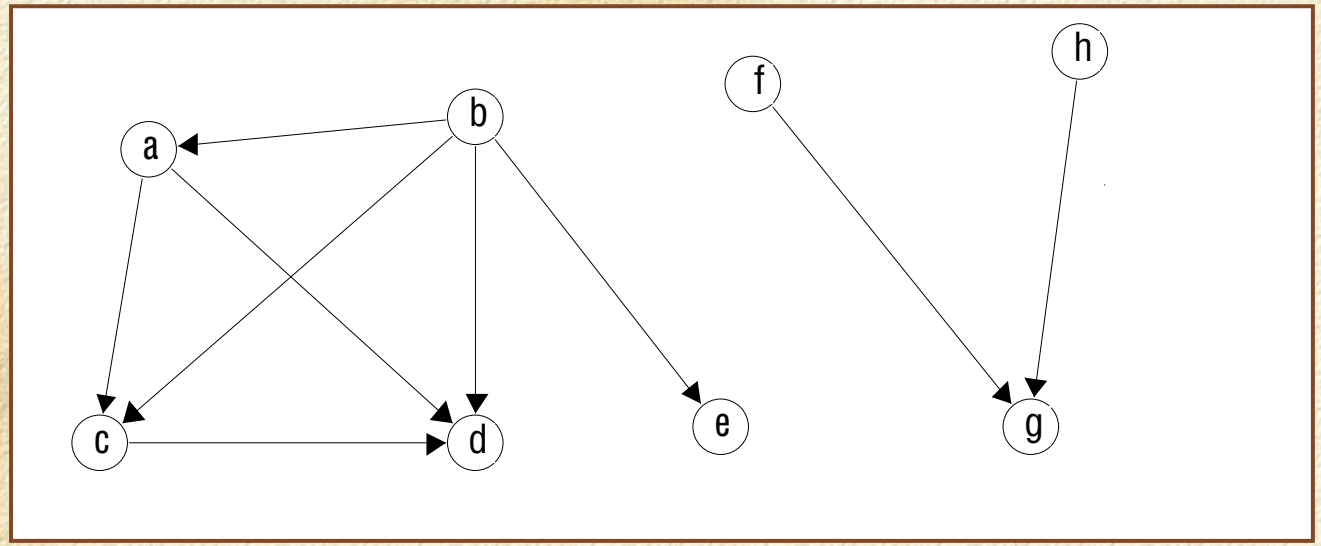
$$R = \{(a,a),(a,c),(b,a),\ldots\}$$





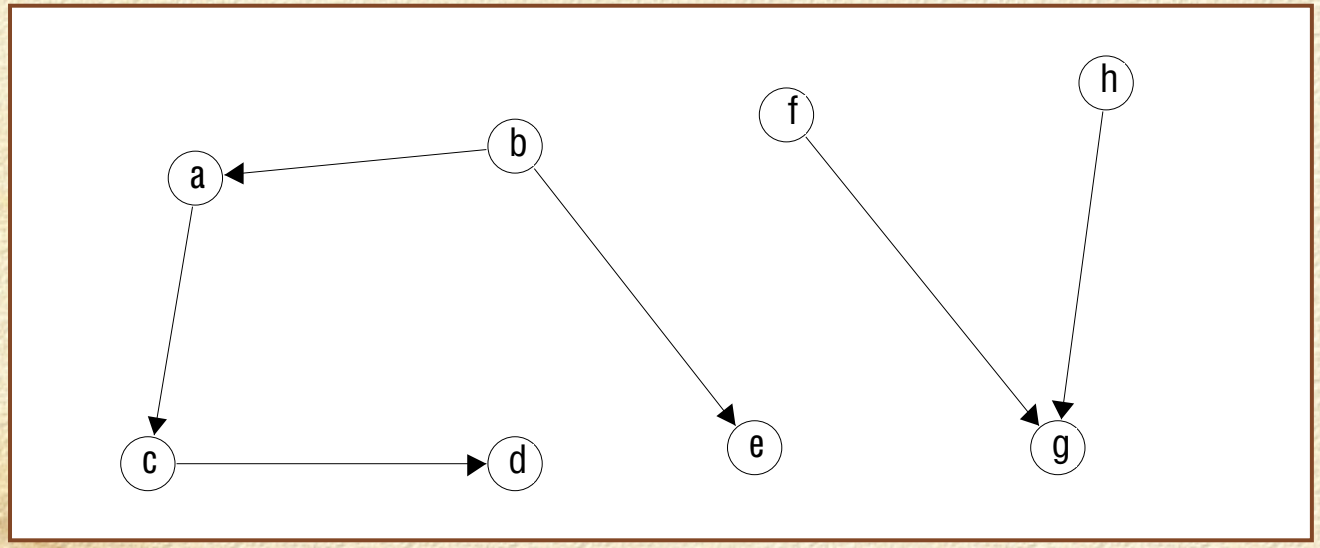
strikte Ordnung

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

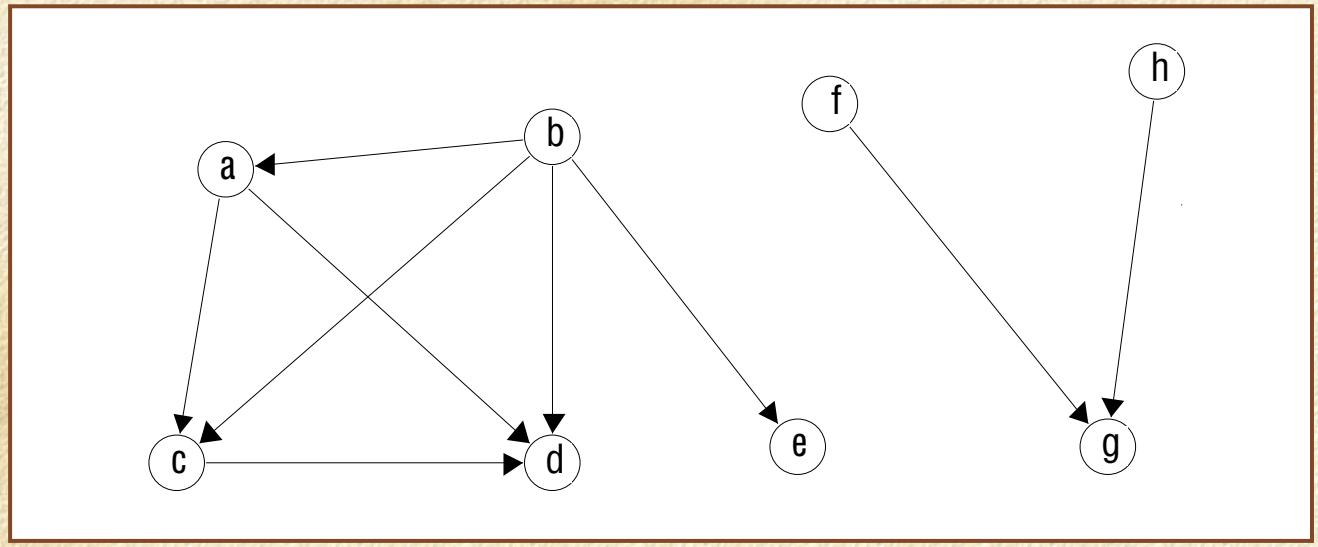


strikte Ordnung

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



Präzedenz-Relation



strikte Ordnung

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

Definition 2.3

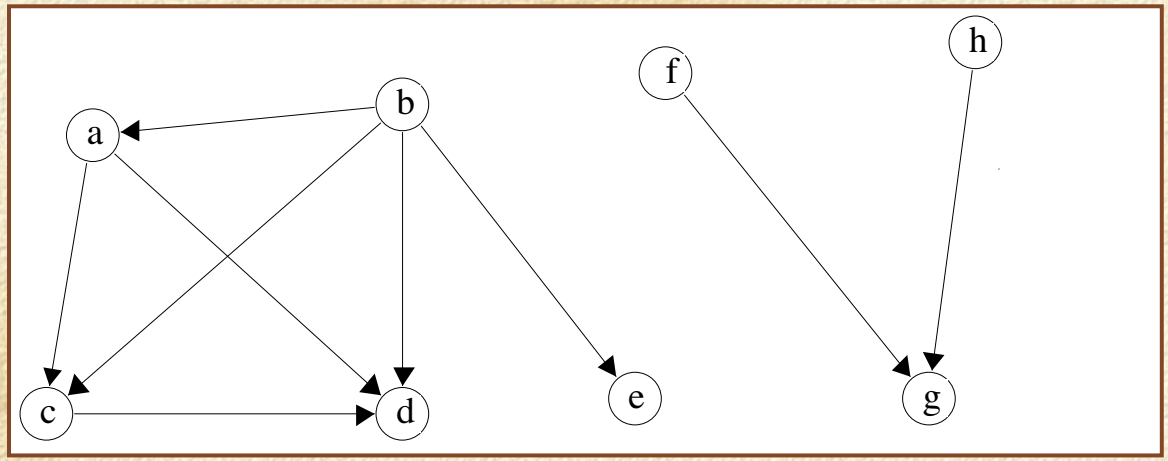
1. *b heißt* direkter Nachfolger *von a (in Zeichen: $a \lessdot b$), falls:*

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

keiner dazwischen

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

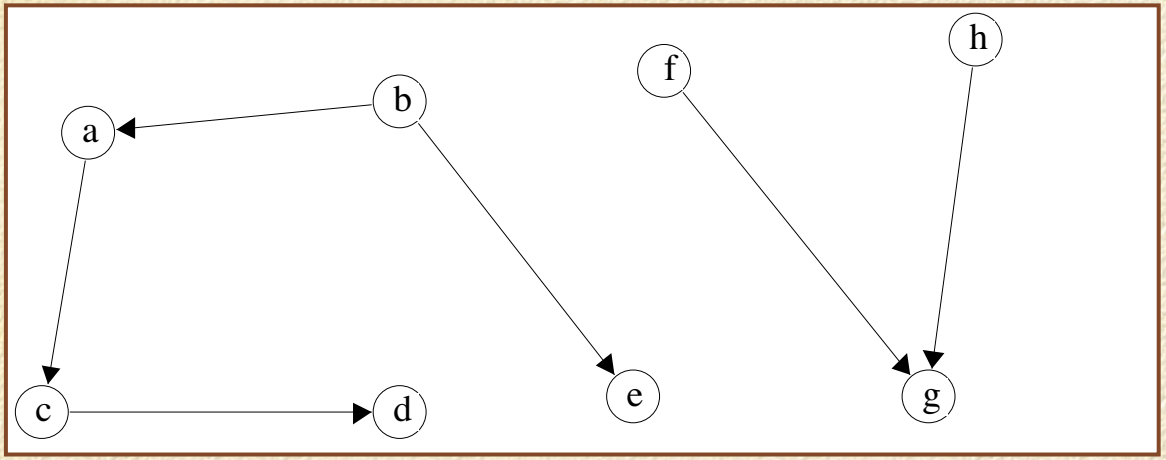
Präzedenz-Relation



strikte Ordnung

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

transitive Hülle



Präzedenz-Ordnung

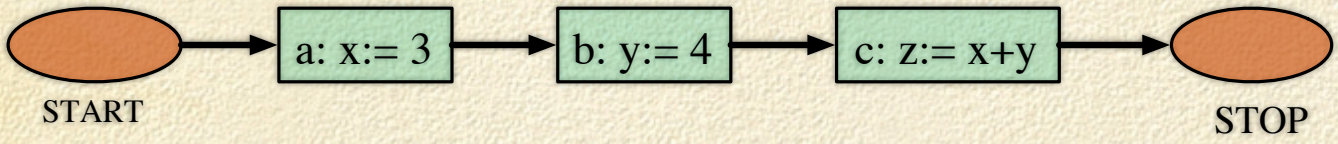
$$Q = S - S^2 = \{(b.a), (a,c), (c,d), \cancel{(a,d)}, \cancel{(b,c)}, \cancel{(b,d)}, (b,e), (f,g), (h,g)\}$$

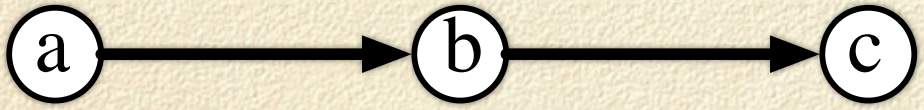
$$<^2 := < \circ <= \{(a,b) \mid \exists c.\ a < c \wedge c < b\}$$

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

1. (A, R) *ist partielle Ordnung*
2. $\forall a, b \in A.\ a \neq b$ *impliziert* $(a, b) \in R \vee (b, a) \in R$

"Vollständigkeit"

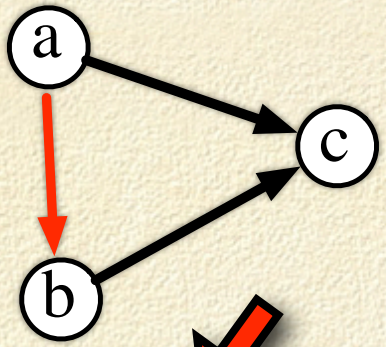




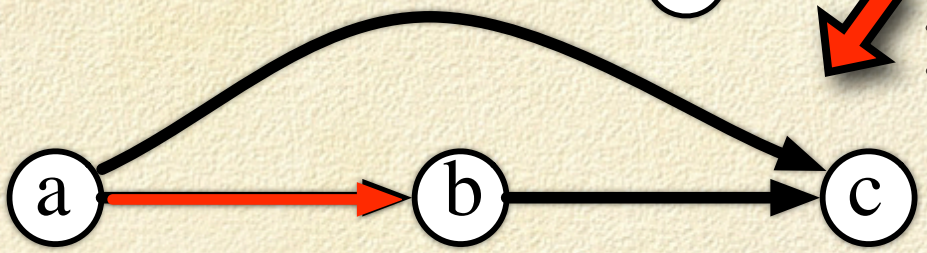
a;b;c und b;a;c

interleaving semantics

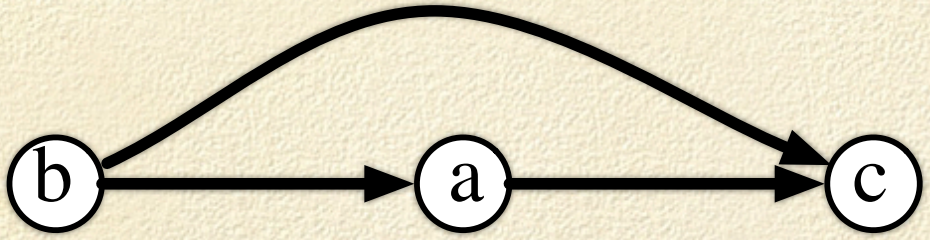
Folgen-Semantik



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



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



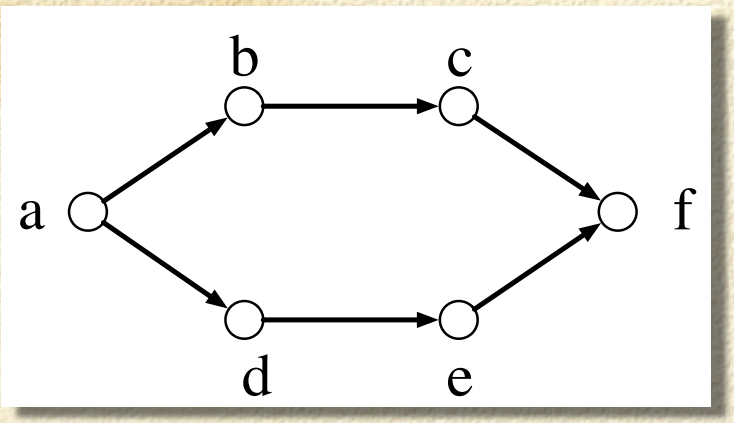
{(b,a),(a,c),(b,c)}

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 $(A, <)$.

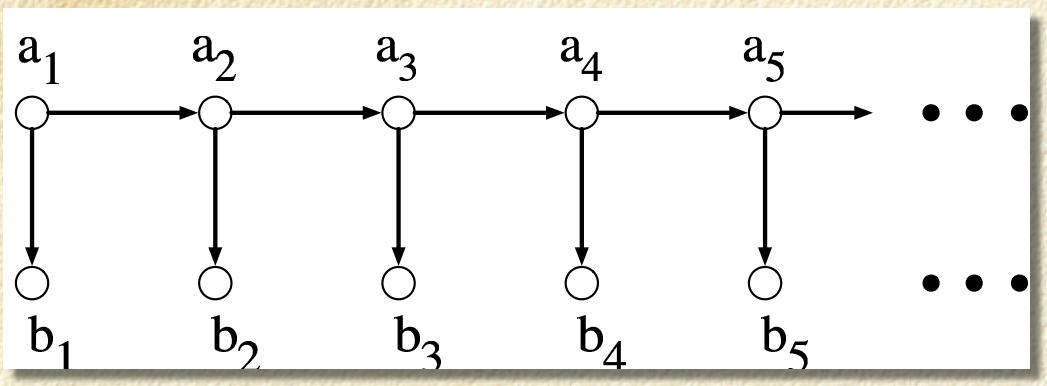
Beispiel: $(A, \lessdot)$



$$Lin(A, \lessdot) = \{\ a\ b\ c\ d\ e\ f\ ,\ a\ b\ d\ c\ e\ f\ ,\ a\ d\ b\ c\ e\ f\ ,\ a\ d\ b\ e\ c\ f\ ,\ a\ d\ e\ b\ c\ f\ ,\ a\ b\ d\ e\ c\ f\ \}$$

Konstruktionsprinzip?

Beispiel: $(A, \lessdot)$



$$Lin(A, \lessdot) = \{\ a_1\ b_1\ a_2\ b_2\ a_3\ b_3\ a_4\ b_4\ a_5\ b_5\ \ldots$$
$$a_1\ a_2\ b_1\ b_2\ a_3\ b_3\ a_4\ b_4\ a_5\ b_5\ \ldots$$
$$a_1\ a_2\ b_1\ a_3\ b_2\ b_3\ a_4\ b_4\ a_5\ b_5\ \ldots$$
$$\vdots$$

Konstruktionsprinzip?

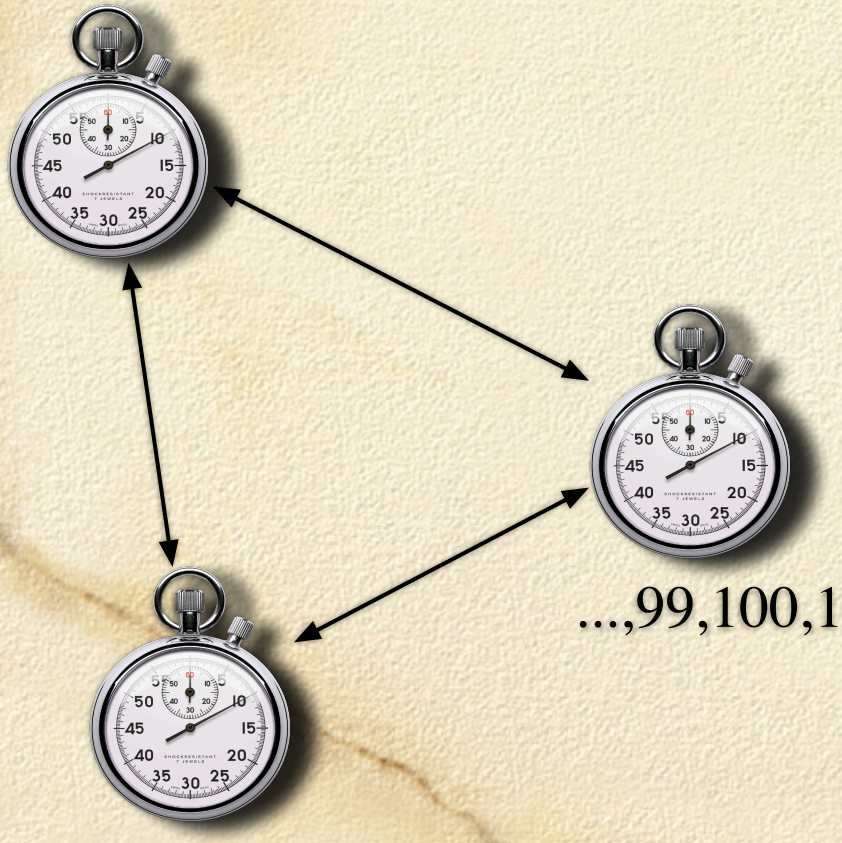
2.2 Logische und vektorielle Zeitstempel

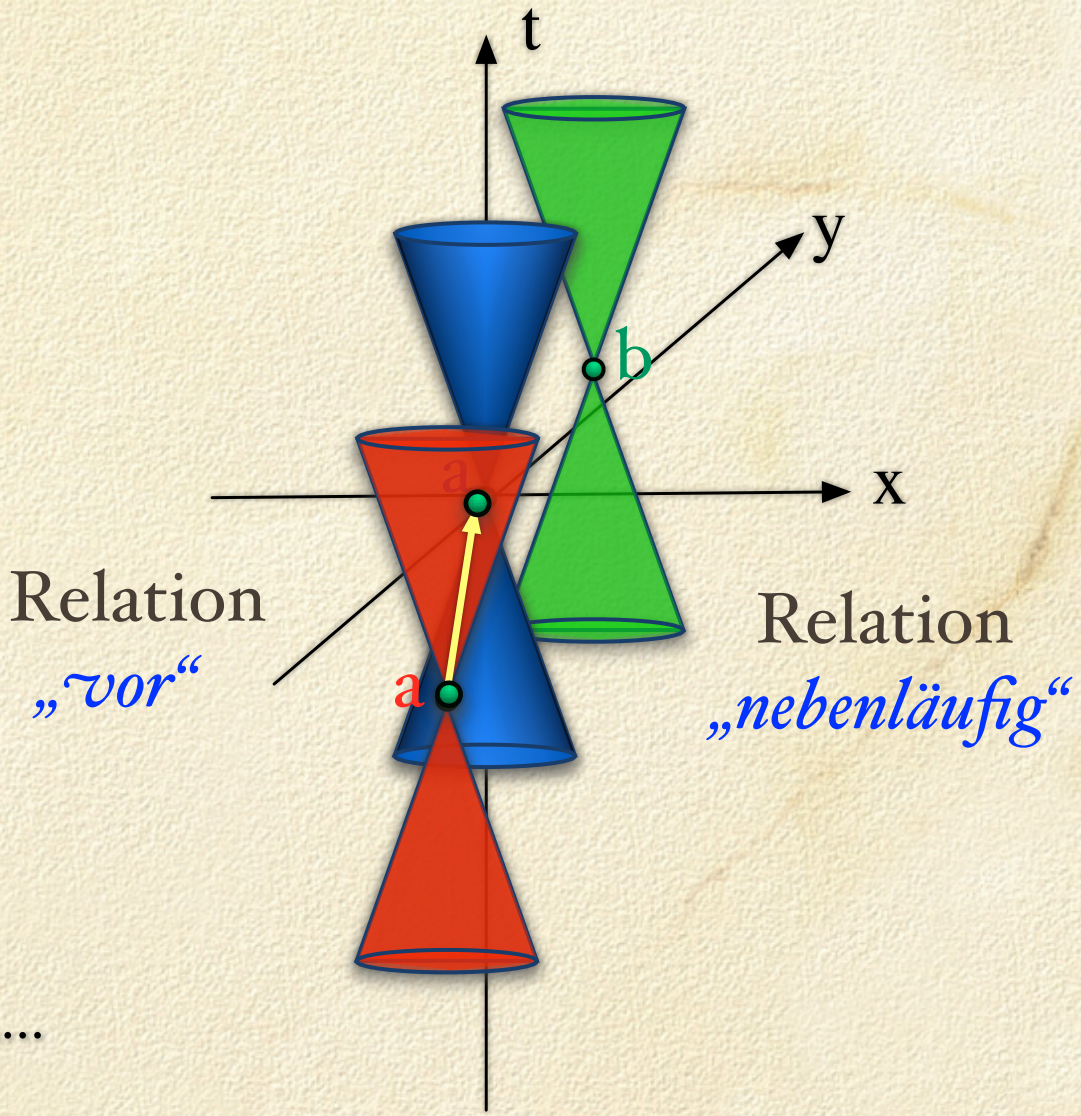
globale Zeit

vs.

lokale Zeit

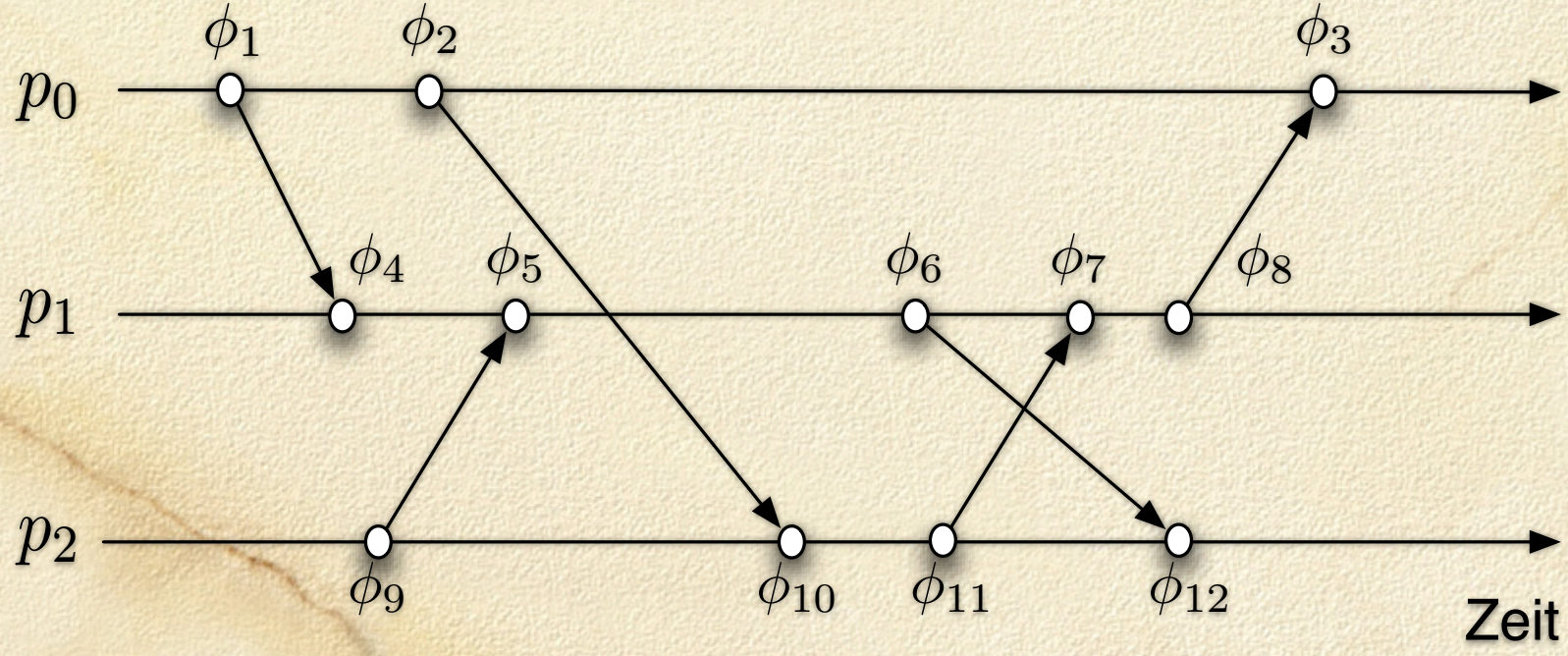
Minkowski-Diagramm wurde 1908 von Hermann Minkowski entwickelt





Definition 2.5 *Ein* Nachrichten-Modell *ist ein System von n Funktionseinheiten bzw. Prozessoren $p_0, \ldots, p_{n-1}$, die*

- *lokale Rechenschritte ausführen und*
- *Nachrichten an andere versenden.*



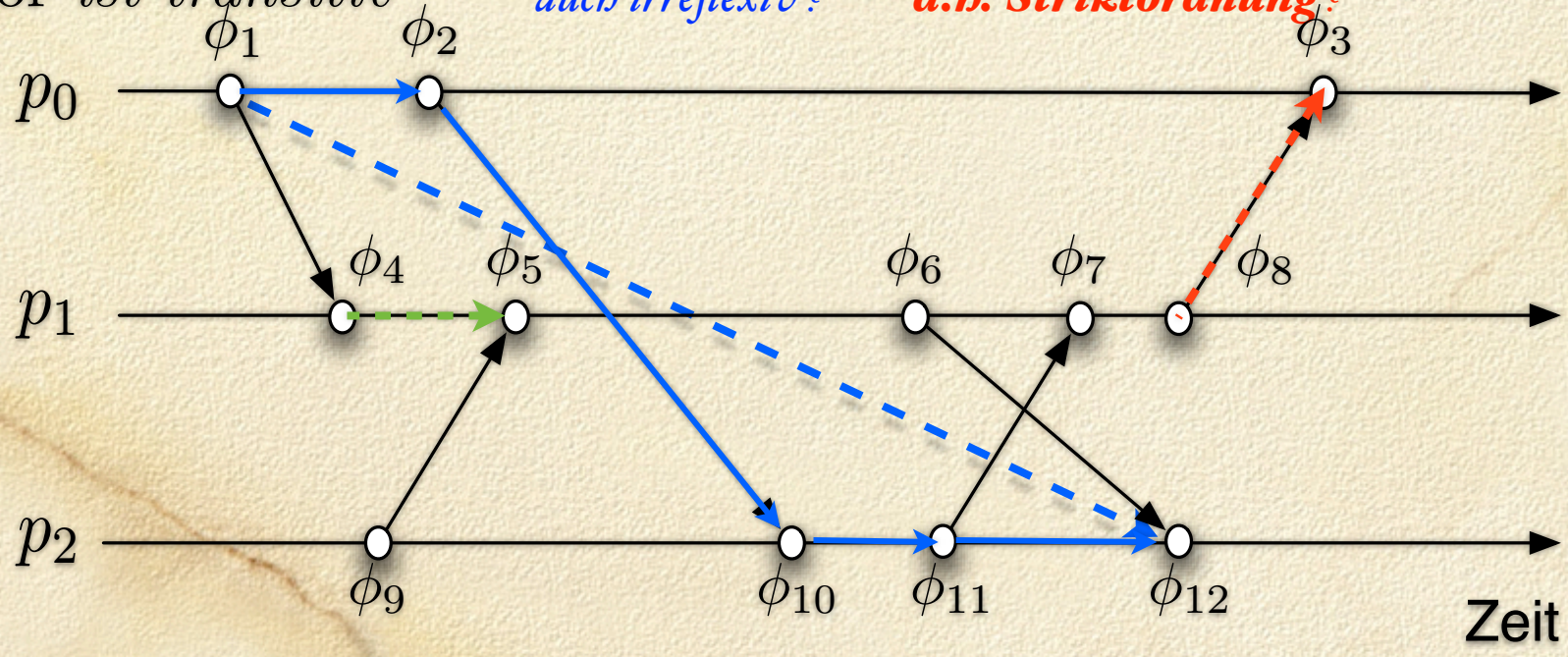
Definition 2.6 *Es wird eine Relation* vor $\subseteq \Phi \times \Phi$ *definiert. Für* $\phi_1, \phi_2 \in \Phi$ *gelte* $(\phi_1$ *vor* $\phi_2)$*, falls folgendes gilt:*

a) Gehören ϕ_1 *und* ϕ_2 *zu dem selben Prozessor (d.h. sie liegen auf der selben (linear geordneten) Zeitachse), dann gilt* $(\phi_1$ **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* $(\phi_1$ vor $\phi_2)$.

c) vor *ist transitiv*

auch irreflexiv ? ***d.h. Striktordnung ?***

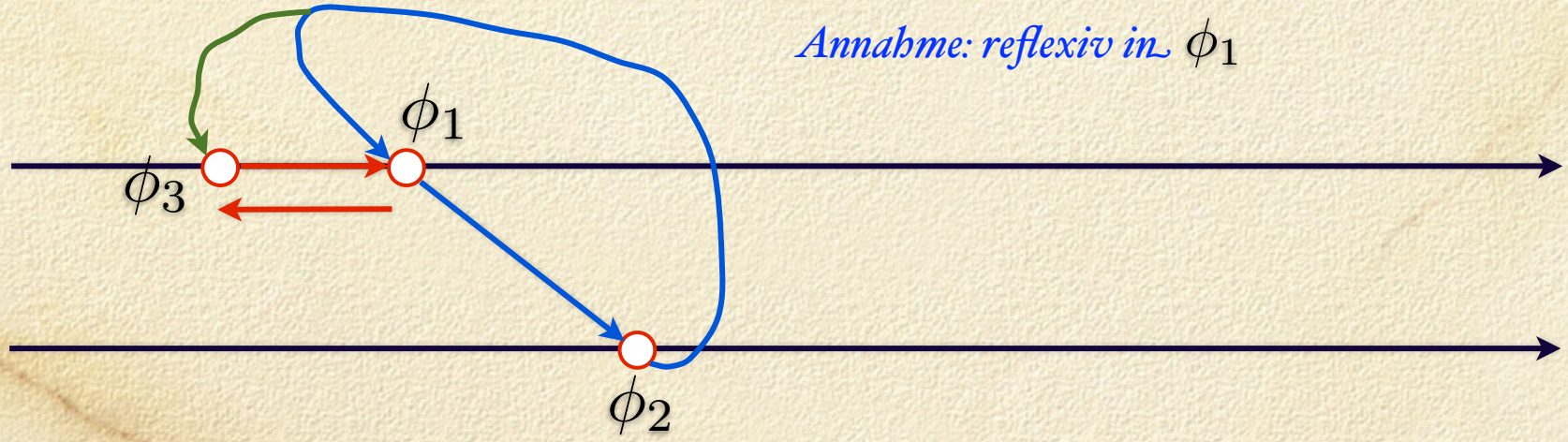


Definition 2.6 *Es wird eine Relation* vor $\subseteq \Phi \times \Phi$ *definiert. Für* $\phi_1, \phi_2 \in \Phi$ *gelte* (ϕ_1 *vor* ϕ_2), *falls folgendes gilt:*

a) Gehören ϕ_1 *und* ϕ_2 *zu dem* selben *Prozessor (d.h. sie liegen auf der selben Zeitachse), dann gilt* (ϕ_1 vor ϕ_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* (ϕ_1 vor ϕ_2).

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



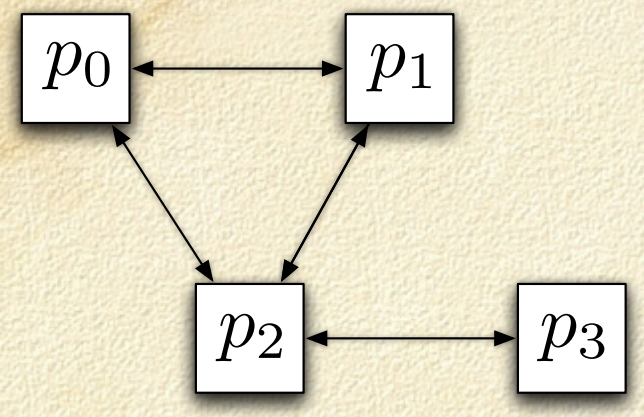
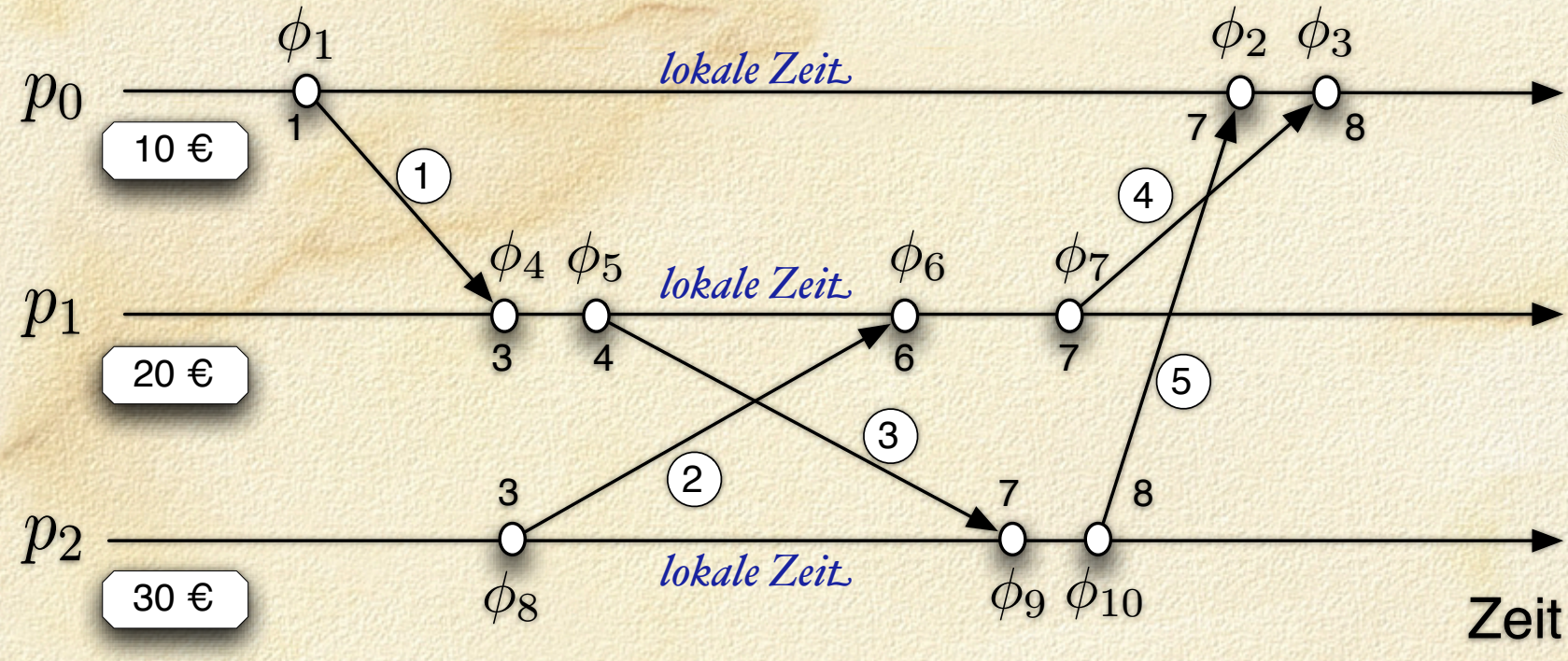


Abbildung 2.5 Banksystem mit 4 Filialen $p_0, \ldots p_3$

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

Abbildung 2.6 Übertragung von Nachrichten über Kanal

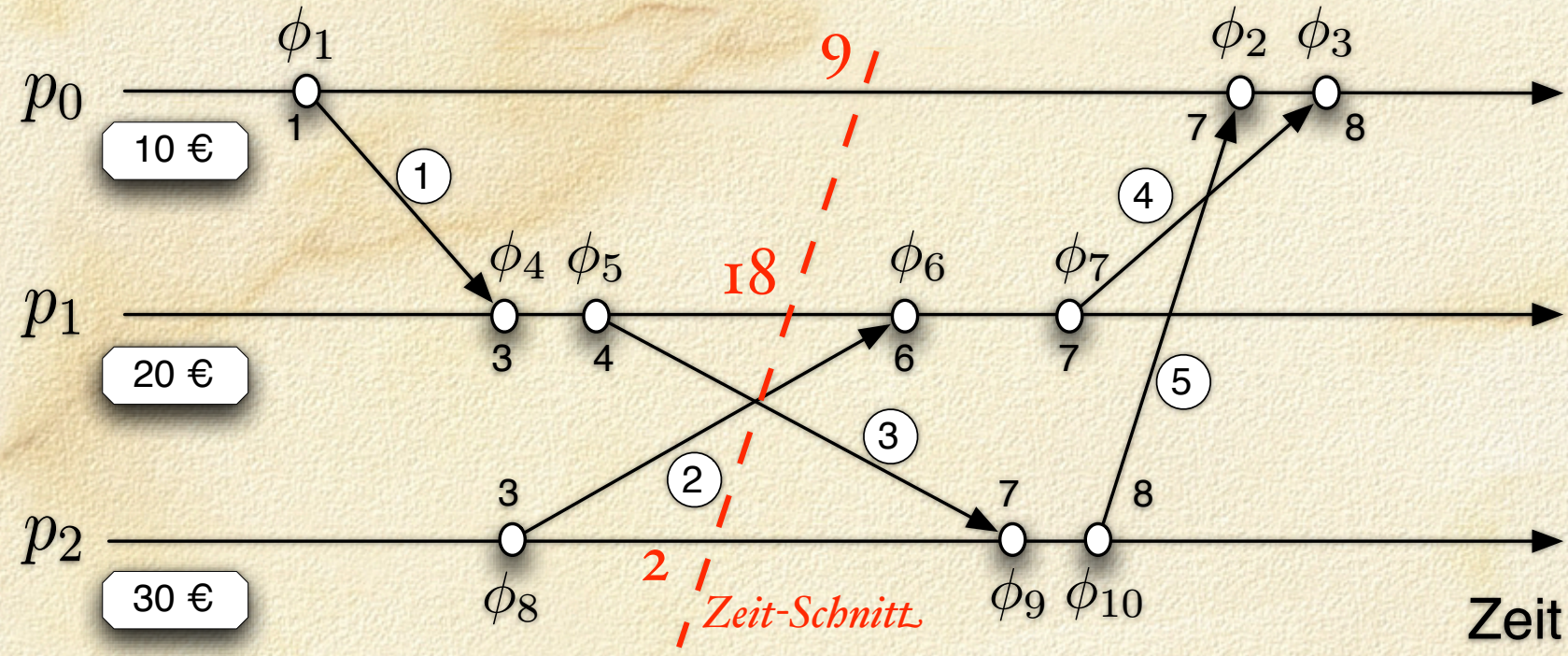


Gesamtgeldmenge : $10 + 20 + 30 = 60$

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

$1, 2, 3, \ldots, 10$ 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 ;

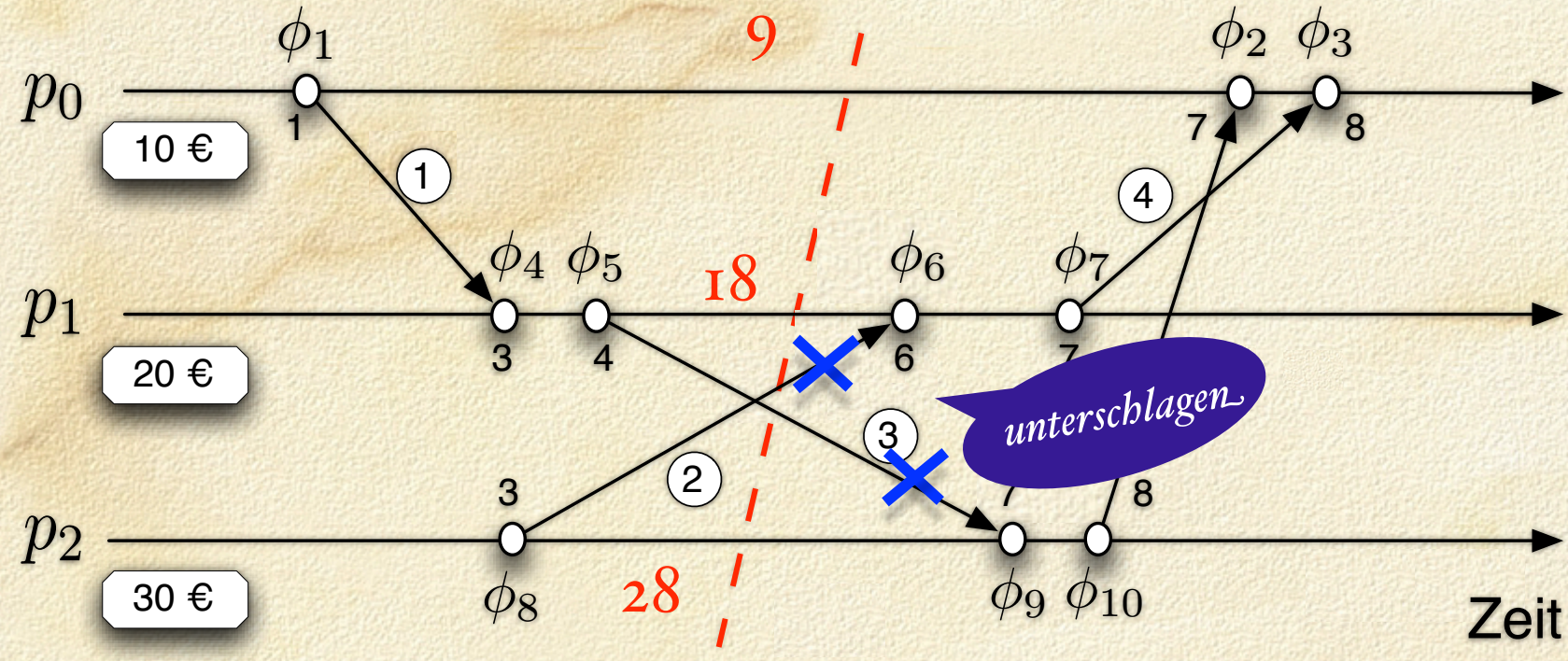
Beispiel 2.8 Zeitpunkt $t = 5$

p_0	:	$x_0 =$	$10 - 1$	$= 9$
p_1	:	$x_1 =$	$20 + 1 - 3$	$= 18$
p_2	:	$x_2 =$	$30 - 2$	$= 28$
			$\sum$	**55 !**

unterwegs: 5 €

also Problem:

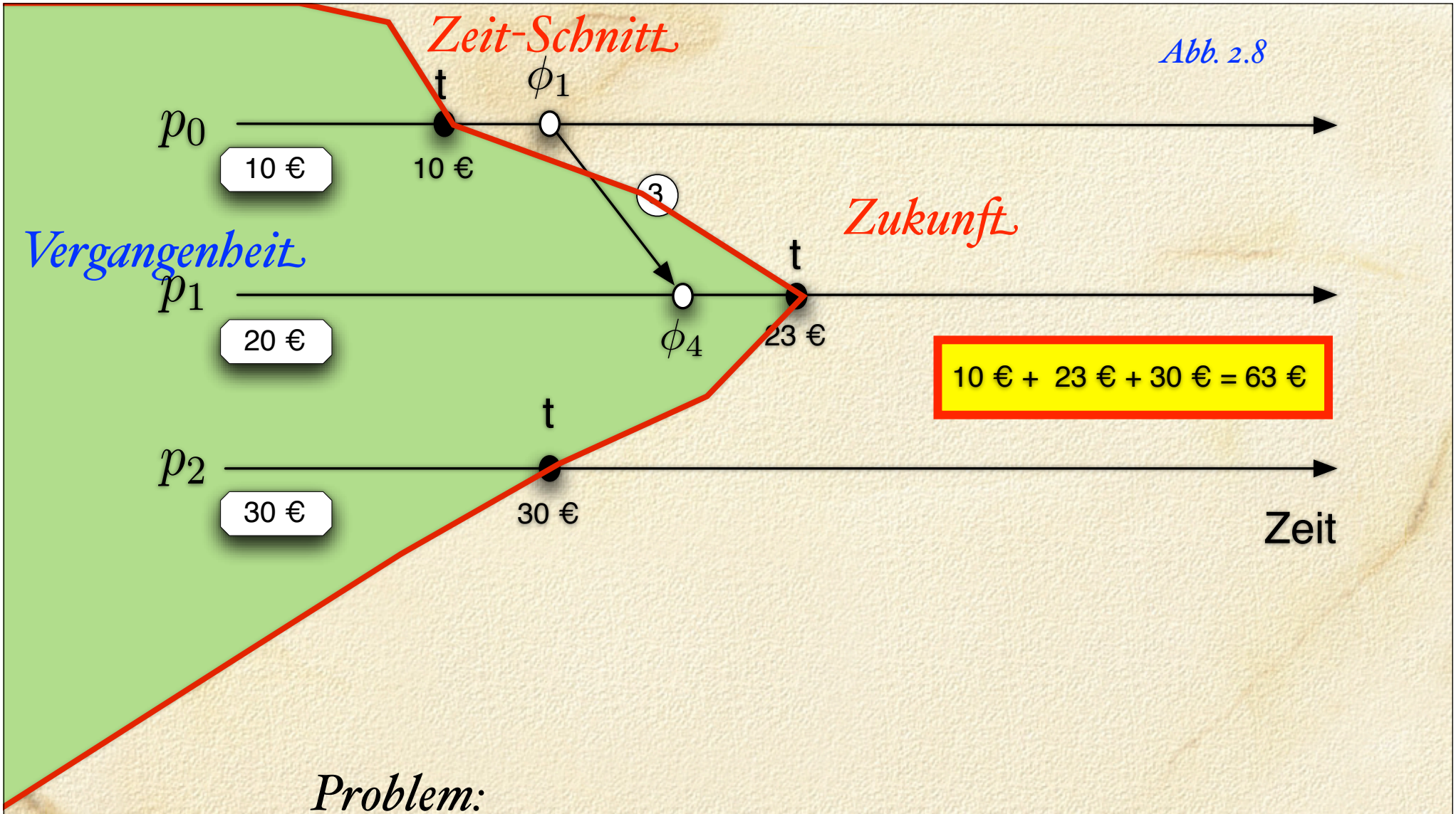
die noch nicht empfangenen Nachrichten!



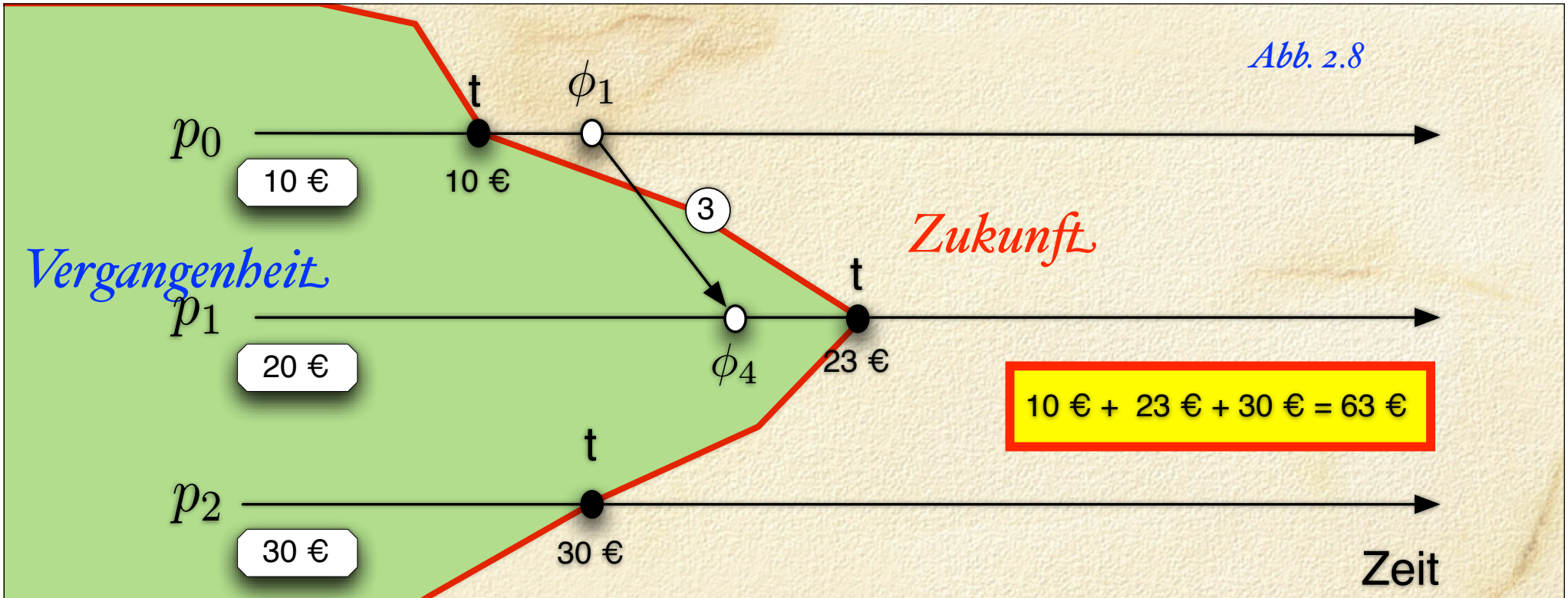
Verfahren 2:
Die Bankleitung bittet die Summe der abgesandten minus der Summe der eingegangenen Beträge zu x_i jeweils hinzuzuzählen.
Erwartung : $\sum = 60$.

Beispiel 6.18 Zeitpunkt $t = 5$

p_0	: $x_0 =$	$9 + 1$	$=$	10
p_1	: $x_1 =$	$18 - 1 + 3$	$=$	20
p_2	: $x_2 =$	$28 + 2$	$=$	30
			$\sum$	**60**



Problem:
Nachrichten aus der Zukunft in die Vergangenheit !



Deshalb stellt sich die Frage, wie die Relation *vor im* System beobachtet werden kann. Dazu bestimmen wir eine *logische Uhr* $LT(\phi)$ mit

$$\phi_1 \; vor \; \phi_2 \Rightarrow LT(\phi_1) < LT(\phi_2).$$

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.

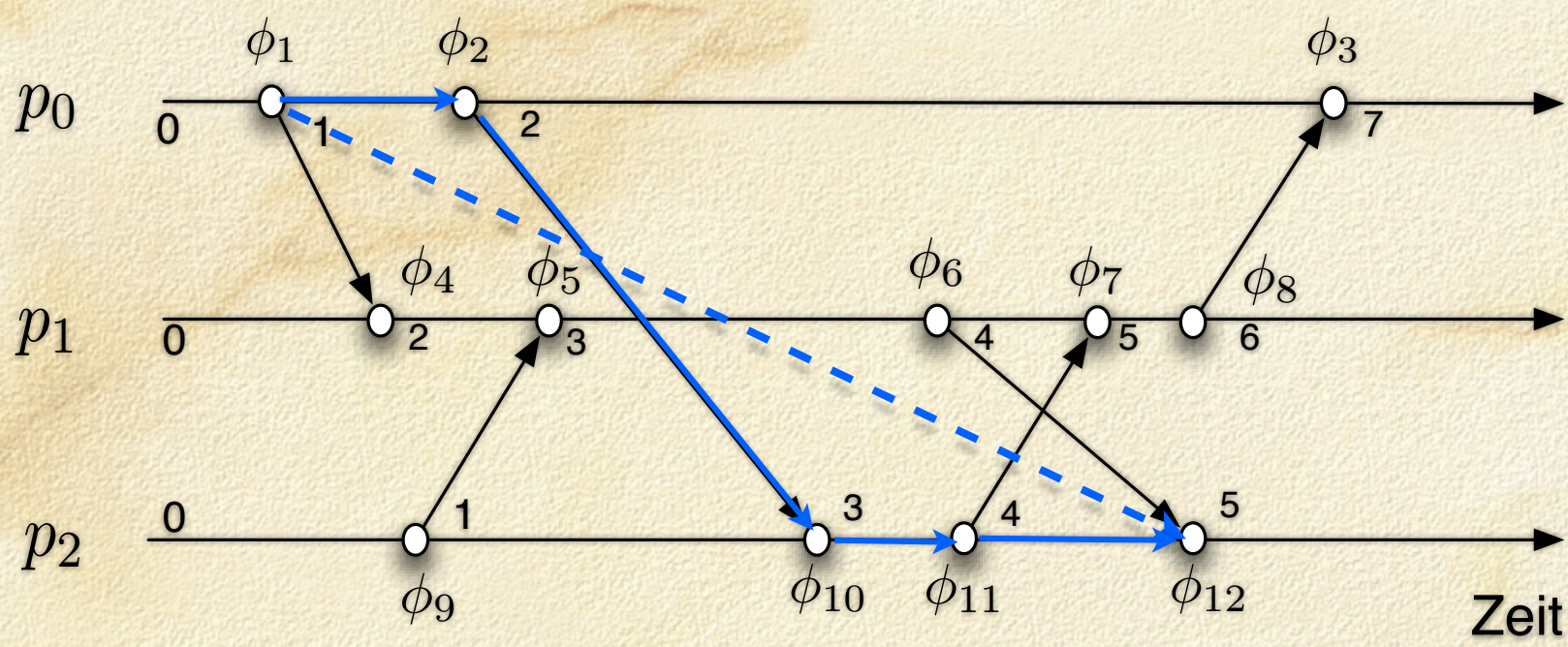


Abbildung 2.9 Senden mit logischen Zeitstempeln

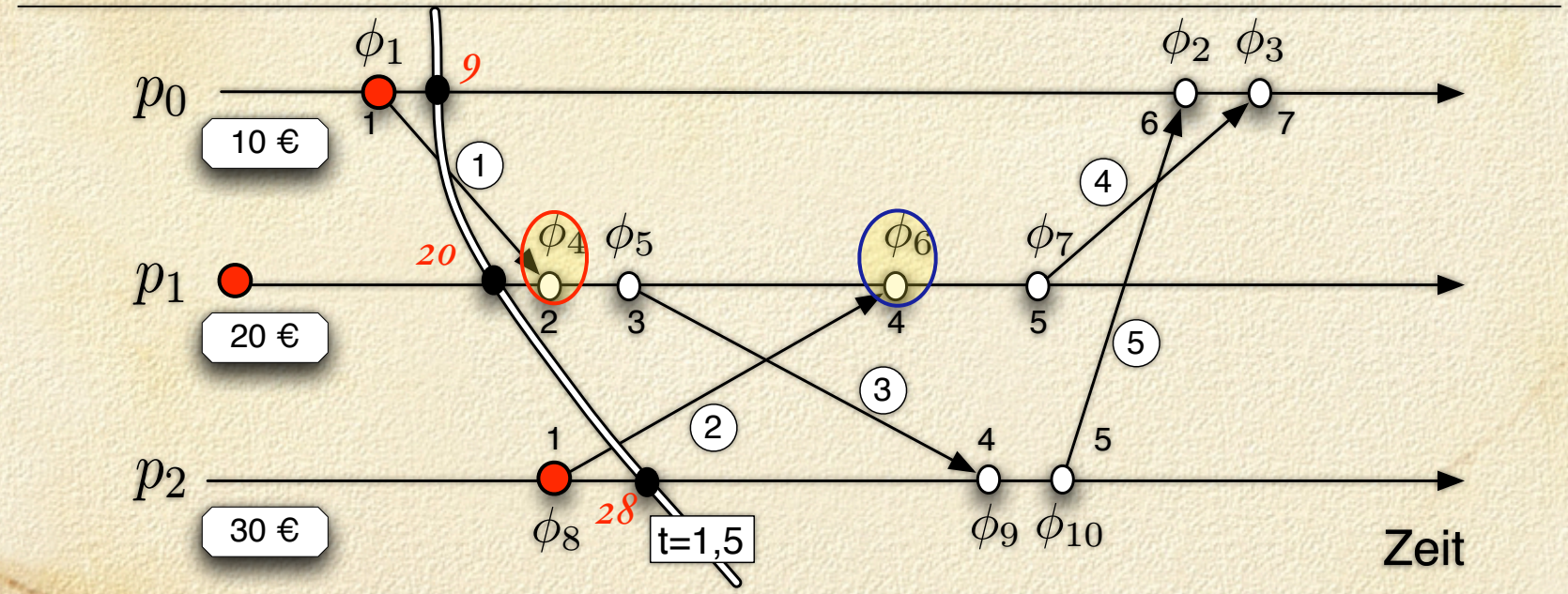
Definition 2.10 *Sei ϕ ein Ereignis von p_i. Dann bezeichnet $LT(\phi)$ den von ϕ berechneten Wert von LT_i.*

Satz 2.11 *Für die Ereignisse $\phi_1, \phi_2 \in \Phi$ gilt :*
ϕ_1 **vor** $\phi_2 \Rightarrow LT(\phi_1) < LT(\phi_2)$

Algorithmus 2.2 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.
   - Sende den Wert jeder Geldsendung an Leitung, die ab ϕ' ankommt, aber einen Zeitstempel $\leq LT(\phi)$ hat.



$9 + 20 + 28 = 57$

Es kommen an : 1 in ϕ_4 und 2 in ϕ_6.

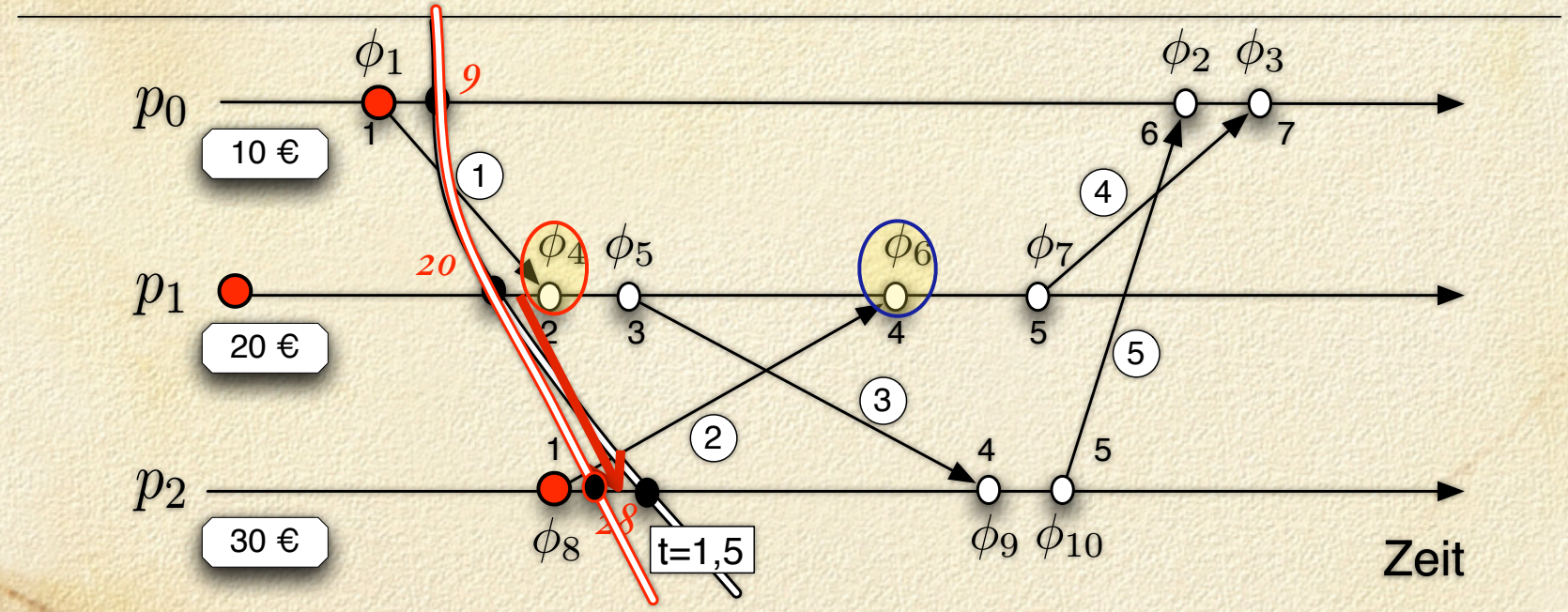
Die Summe ist $57 + 3 = 60$.

Algorithmus 2.2 Verfahren der Bankleitung

1. Man führe logische Uhren ein.

konsistenter Schnitt

Problem:
Nachrichten aus der Zukunft in die Vergangenheit ?

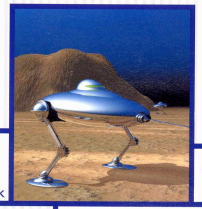


$9 + 20 + 28 = 57$

Es kommen an : 1 in ϕ_4 und 2 in ϕ_6.

Die Summe ist $57 + 3 = 60$.


informatik
Andrew Tanenbaum
Marten van Steen
Verteilte Systeme
Grundlagen und Paradigmen
Prentice Hall
PEARSON
Studium



informatik

Andrew Tanenbaum
Marten van Steen

Verteilte Systeme
Grundlagen und Paradigmen

Prentice Hall

PEARSON Studium

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.

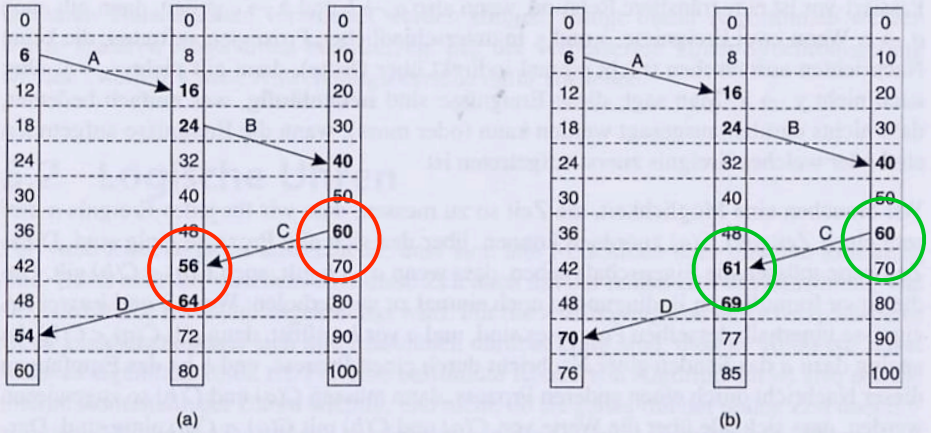
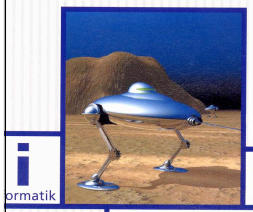


Abbildung 5.7: (a) Drei Prozesse, die jeweils eine eigene Uhr besitzen; die Uhren werden mit unterschiedlichen Geschwindigkeiten ausgeführt. (b) Der Algorithmus von Lamport korrigiert die



Andrew Tanenbaum
Marten van Steen

Verteilte Systeme
Grundlagen und Paradigmen

Pearson Studium

Beispie

Als eine eine
Datenba leis-
tung zu nen
Städten im-
mer an auf
eine Ab ngs-
operation auf jeder Replik ausgeführt werden muss.

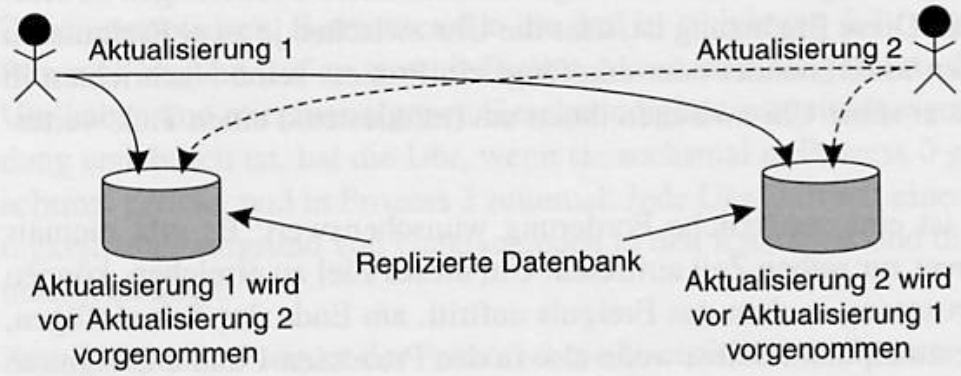
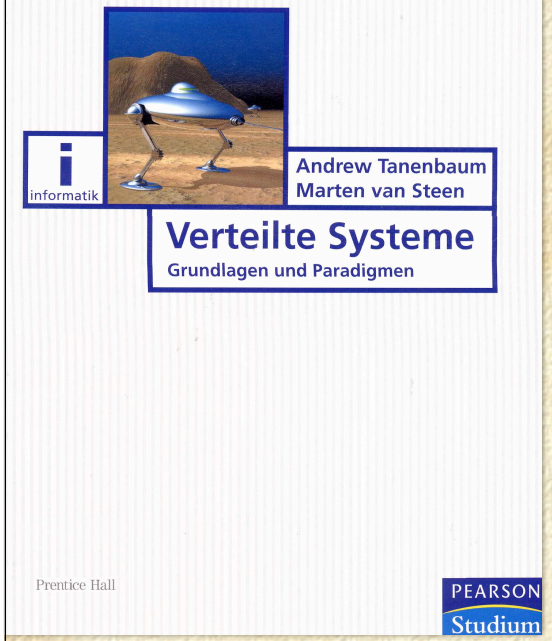


Abbildung 5.8: Aktualisierungen einer replizierten Datenbank und Verursachung eines inkonsistenten Zustands

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.



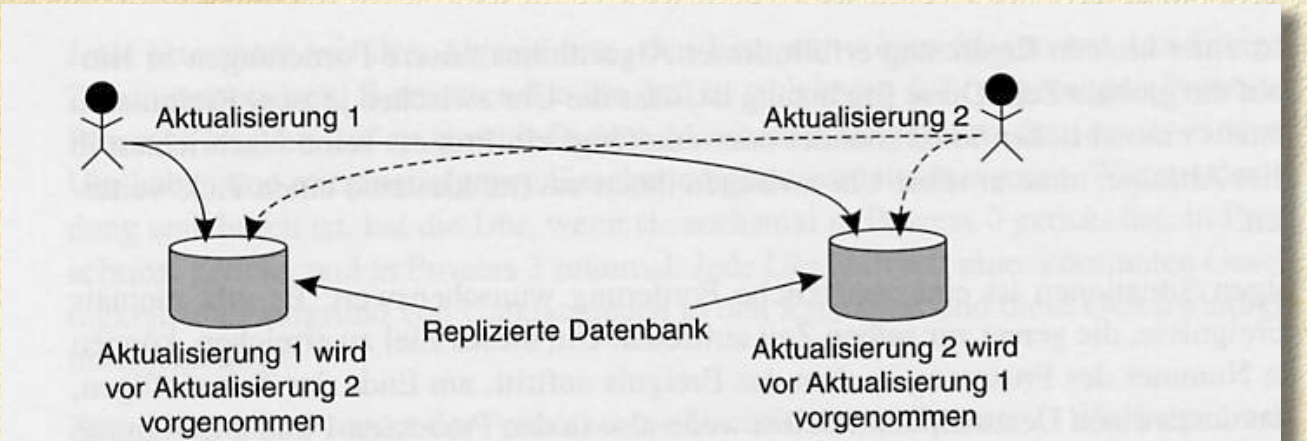
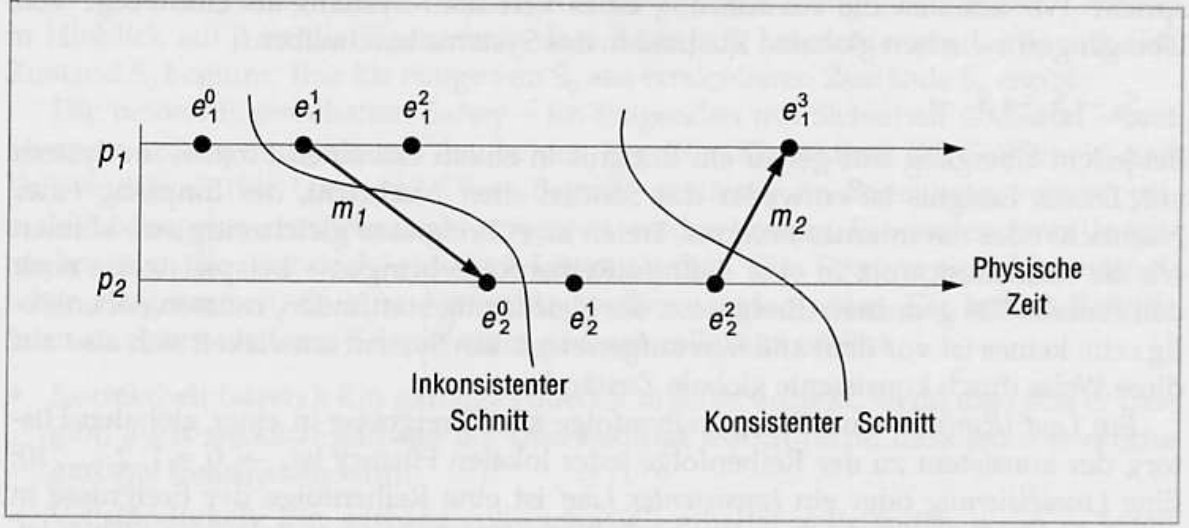


Abbildung 5.8: Aktualisierungen einer replizierten Datenbank und Verursachung eines inkonsistenten Zustands

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.


COMPANION WEBSITE
CWS
www.pearson-studium.de
informatik
George Coulouris
Jean Dollimore
Tim Kindberg
Verteilte Systeme
Konzepte und Design
3., überarbeitete Auflage
ADDISON-WESLEY
Pearson Studium

Abbildung 10.9 Schnitte

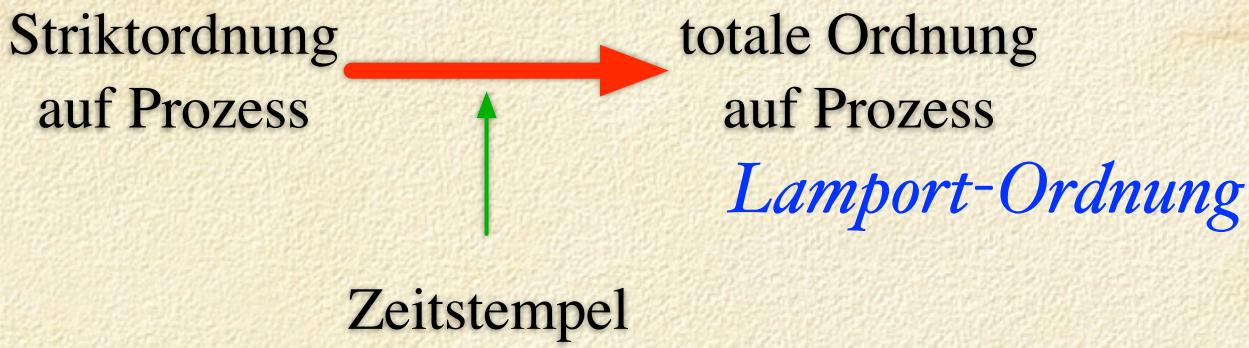


Außerdem können wir die globale History von $\wp$ als die Vereinigung der einzelnen Prozesshistories bilden:

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

Mathematisch können wir eine beliebige Menge mit Zuständen der einzelnen Prozesse heranziehen, um einen globalen Zustand $S = (s_1, s_2, \ldots, s_N)$ zu bilden. Aber wel-

in Anwendungen oft benötigt: totale Ordnung auf Ereignissen im Netz



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

gdw.

$$a) LT(\phi_1) < LT(\phi_2) \quad oder$$

$$b) LT(\phi_1) = LT(\phi_2) \quad und \quad i < j$$

tie-break-rule

$(2, p_5) < (3, p_2)$ $(2, p_5) > (2, p_2)$ $(2, p_5) < (4, p_8)$

L. Lamport

Lamport-Ordnung

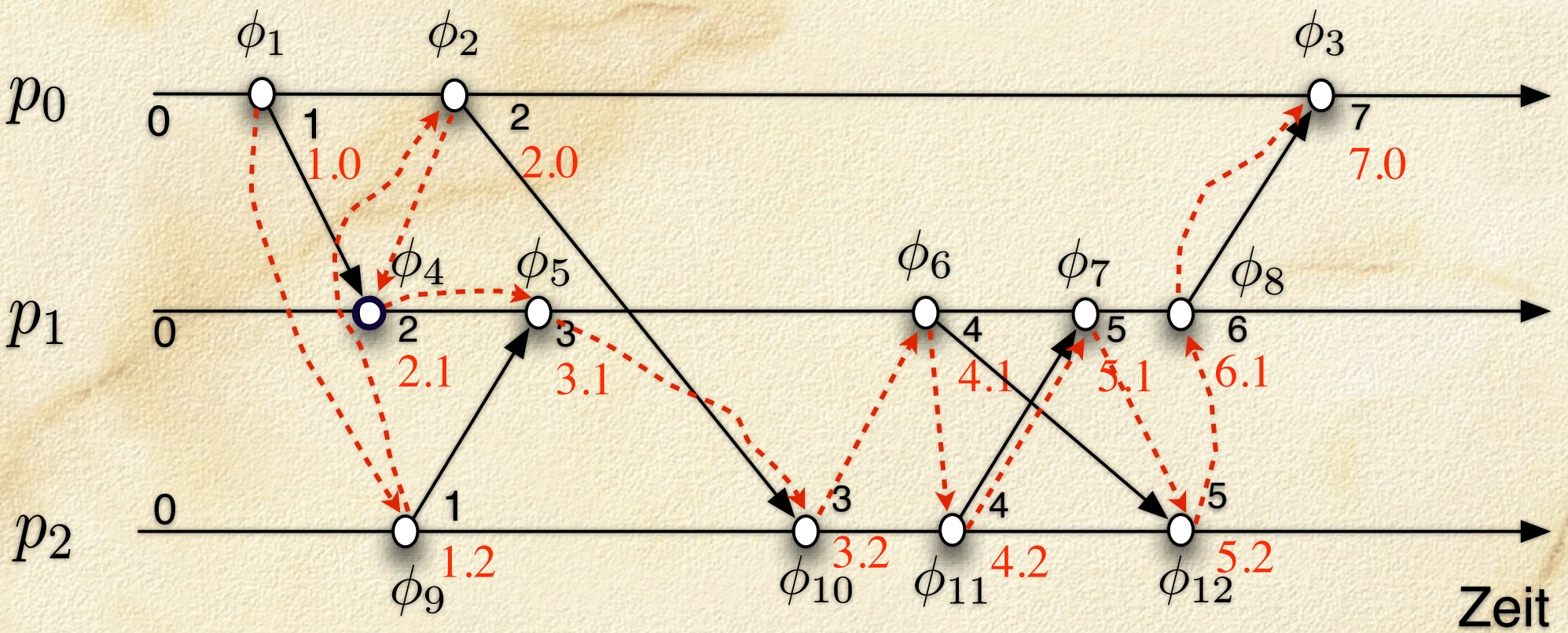
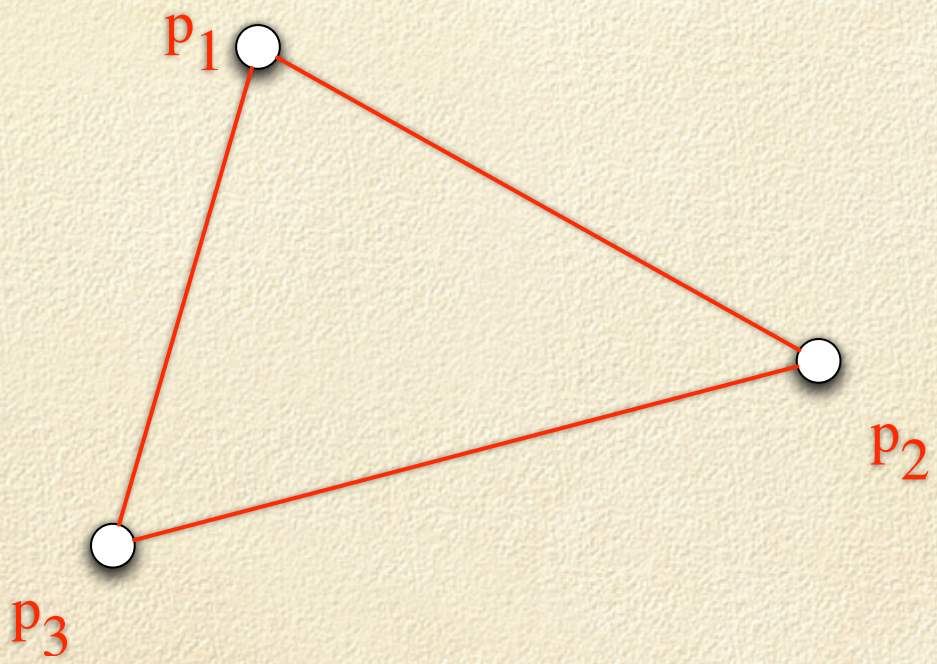
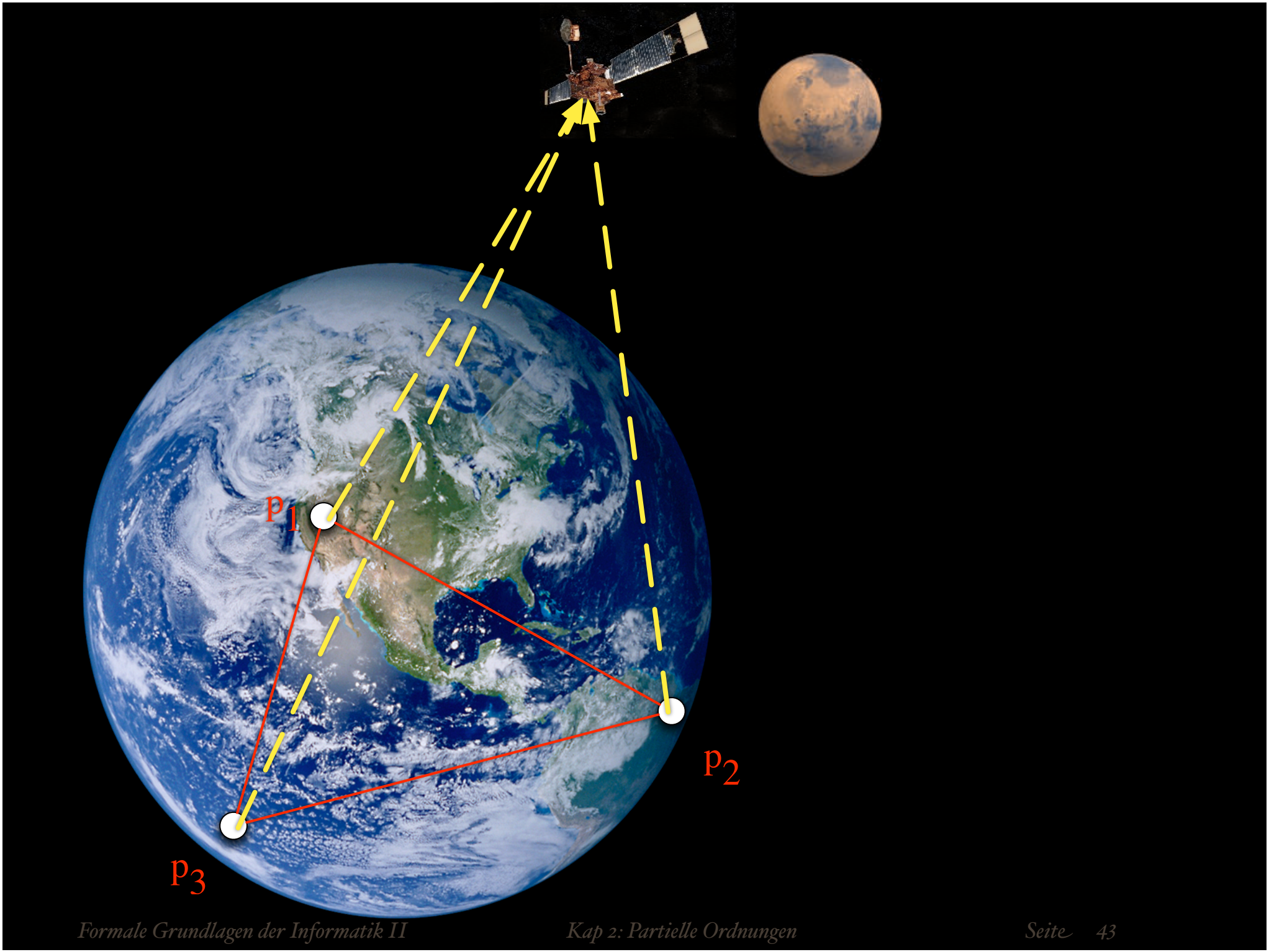


Abbildung 2.9 Senden mit logischen Zeitstempeln

Aufgabe (verteilter wechselseitiger Ausschluss)

Schreiben Sie einen Algorithmus für verteilten wechselseitigen Ausschluss. Dabei senden Prozessoren, die in den kritischen Abschnitt eintreten wollen, eine Anfrage (*request*) an die anderen und legen einen Zeitstempel ts bei. Wenn sie von allen Prozessoren ein *reply* erhalten haben, dürfen Sie in den kritischen Abschnitt eintreten.




p1
p2
p3

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-

Wir haben gesehen, dass gilt:

$$\phi_1 \; vor \; \phi_2 \Rightarrow LT(\phi_1) < LT(\phi_2)$$

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

$$LT(\phi_1) < LT(\phi_2) \Rightarrow \phi_1 \; vor \; \phi_2$$

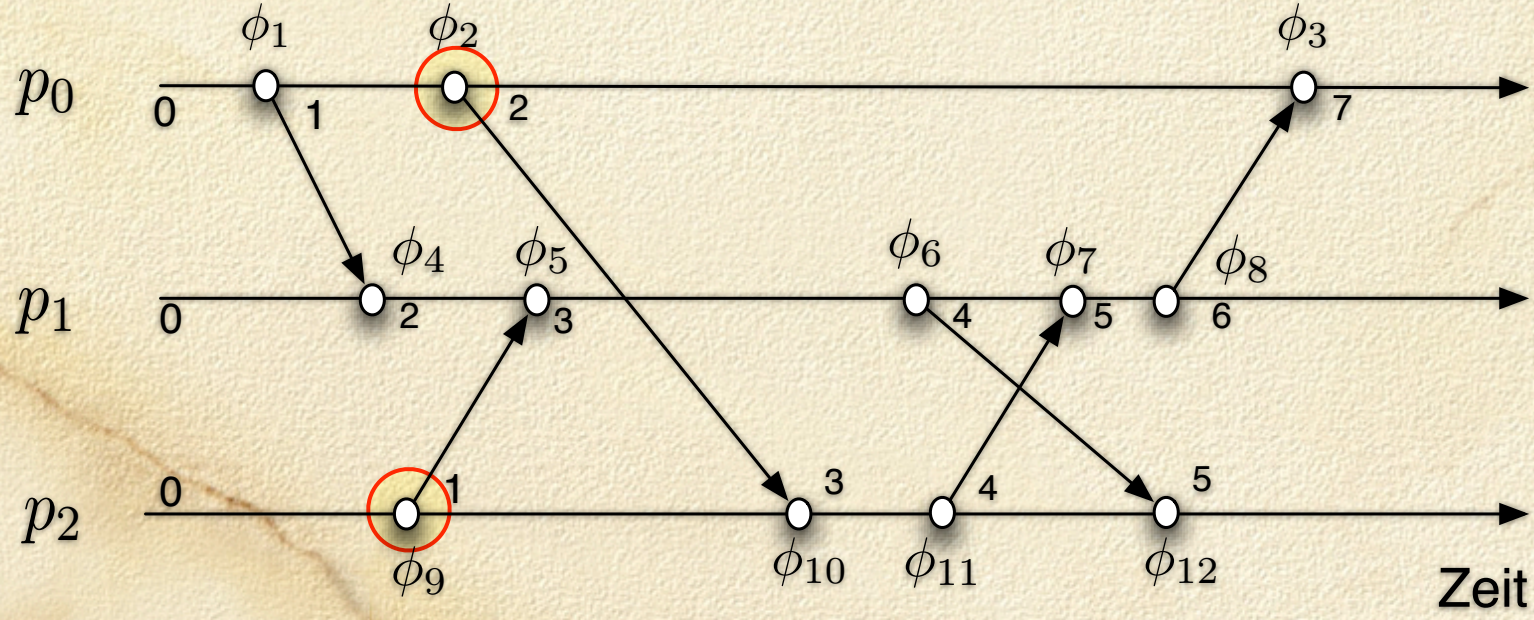
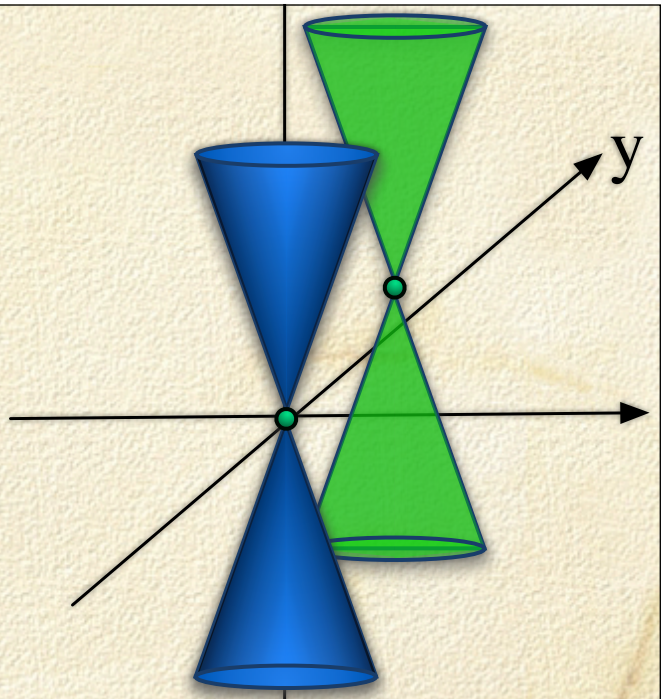


Abb. 2.9

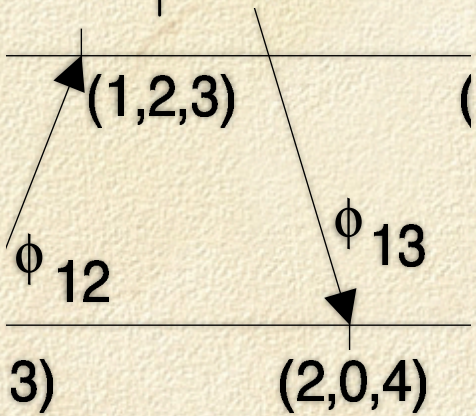
$$LT(\phi_1) < LT(\phi_2) \overset{?}{\Leftarrow} \phi_1 \; vor \; \phi_2$$



Definition 2.15 *ϕ_1 heißt* unabhängig *von ϕ_2 bzw. ϕ_1 heißt* nebenläufig *zu ϕ_2, geschrieben als $\phi_1 \parallel \phi_2$, falls gilt :*

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

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



Definition 2.16 *(Vektorzeit, vektorieller Zeitstempel)*
*Jede Funktionseinheit p_i führt eine Variable $\vec{v}_i$ mit Werten in $\mathbb{N}^n$ (*lokale Vektorzeit*) und dem Nullvektor $\vec{0}$ als Anfangswert. Falls p_i ein Ereignis ϕ bearbeitet, wird der Zeitstempel $\vec{v}_i$ wie folgt aktualisiert:*

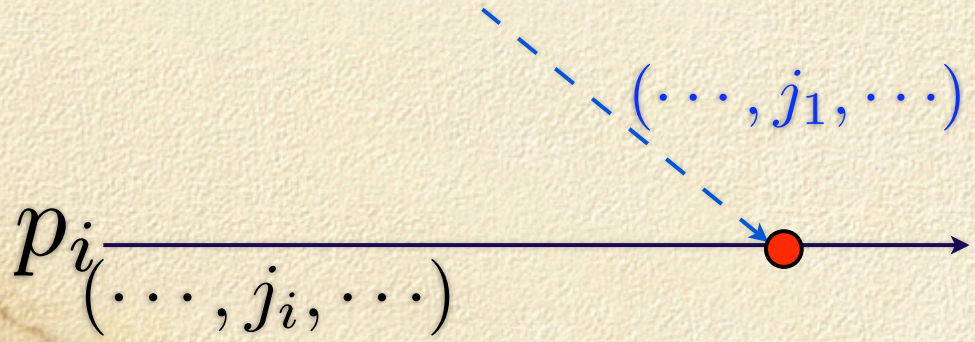
Für die eigene Komponente i von p_i gelte:

$\vec{v}_i[i] \mapsto \vec{v}_i[i] + 1$ *(Inkrementieren des eigenen Stempels)*

Für die anderen Komponenten $j \neq i$ von p_i gelte:

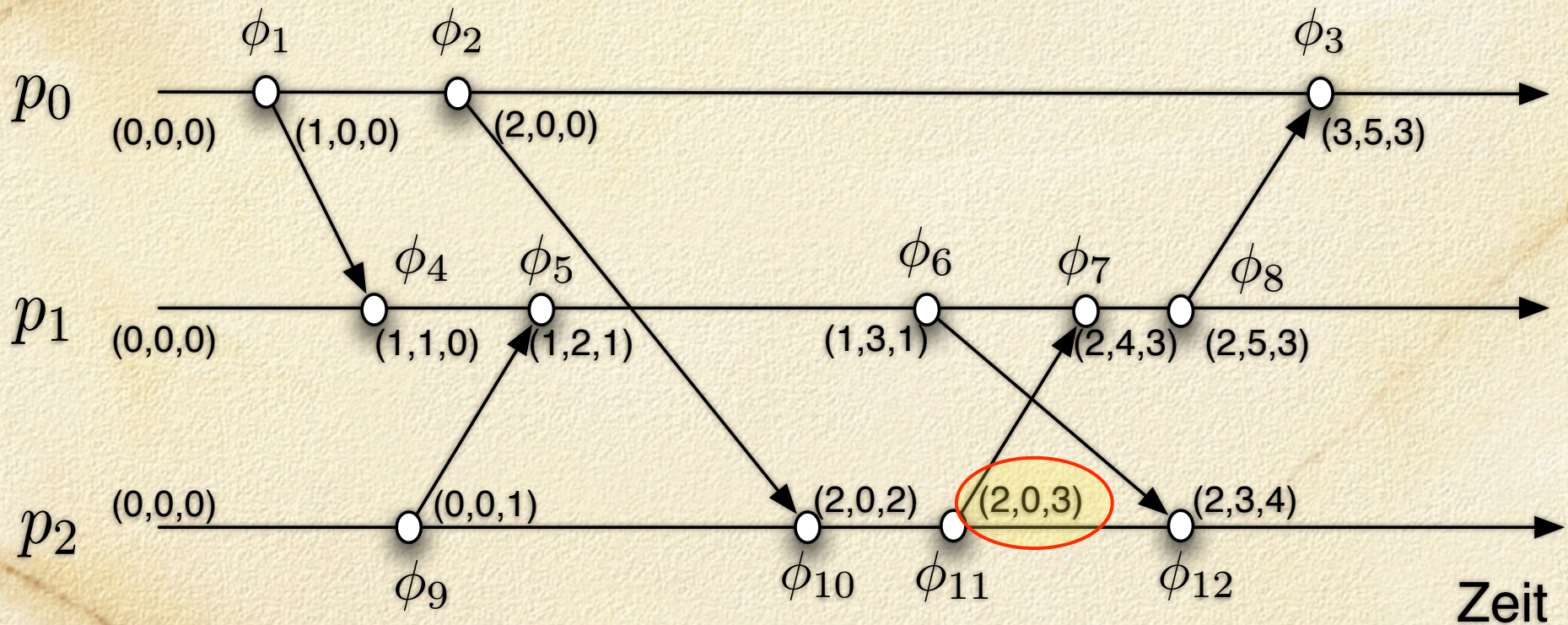
$\vec{v}_i[j] \mapsto \max(\vec{v}_i[j], \vec{VT}_m[j])$ *falls eine Nachricht m mit dem Vektorzeitstempel $\vec{VT}_m$ empfangen wird. Wird keine Nachricht empfangen, bleibt $\vec{v}_i[j]$ unverändert ($j \neq i$).*

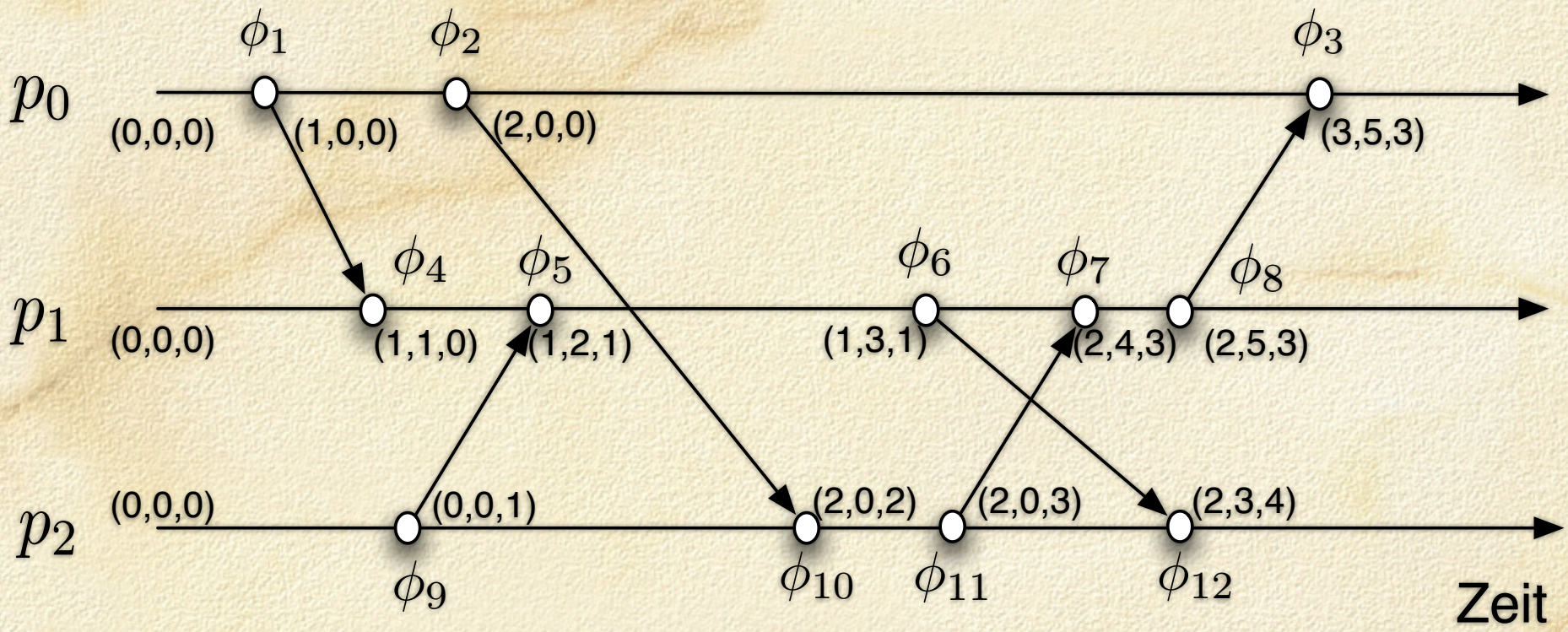
(Aktualisieren der anderen Komponenten)



Definition 2.17 *(Vektor-Uhr)*

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



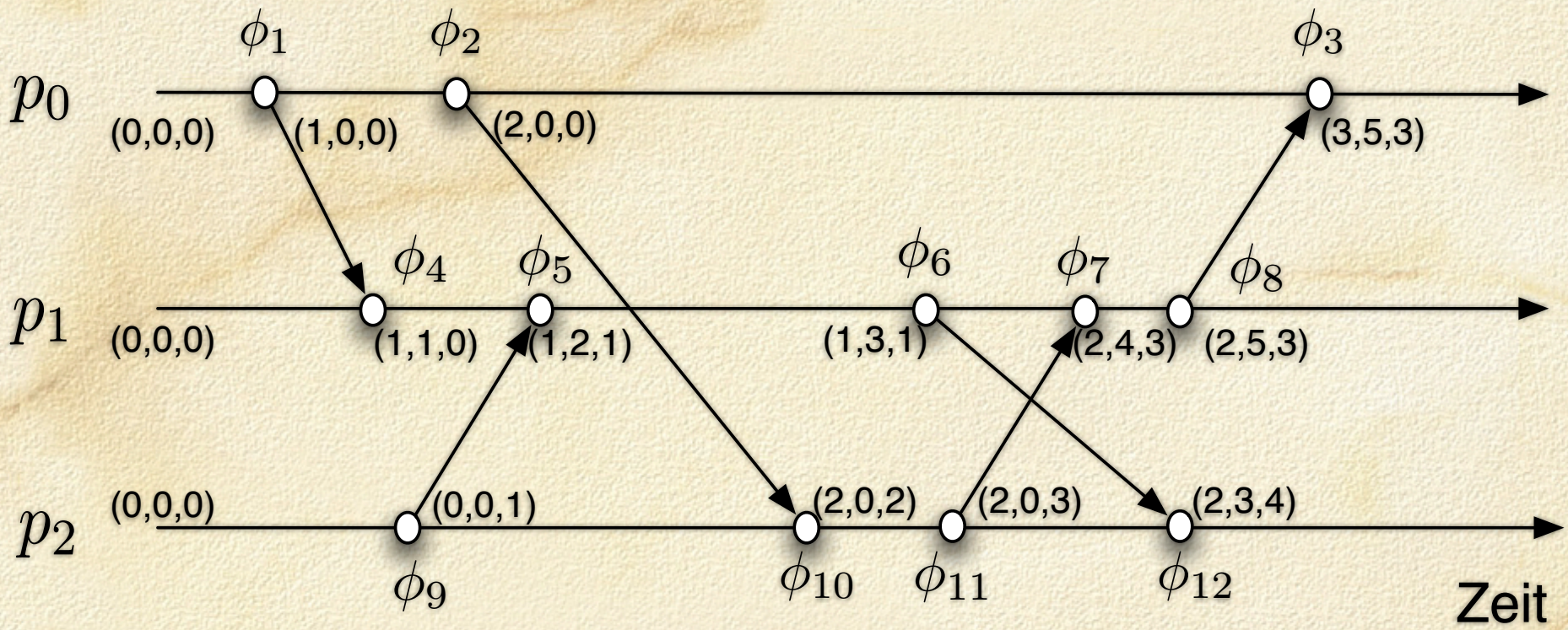


Definition 2.18

a) Partielle Ordnung *auf* $\mathbb{N}^n$ *:* $\vec{v}_1 \leq \vec{v}_2 \Leftrightarrow \forall i \in \{1, \ldots, n\} : \vec{v}_1[i] \leq \vec{v}_2[i]$

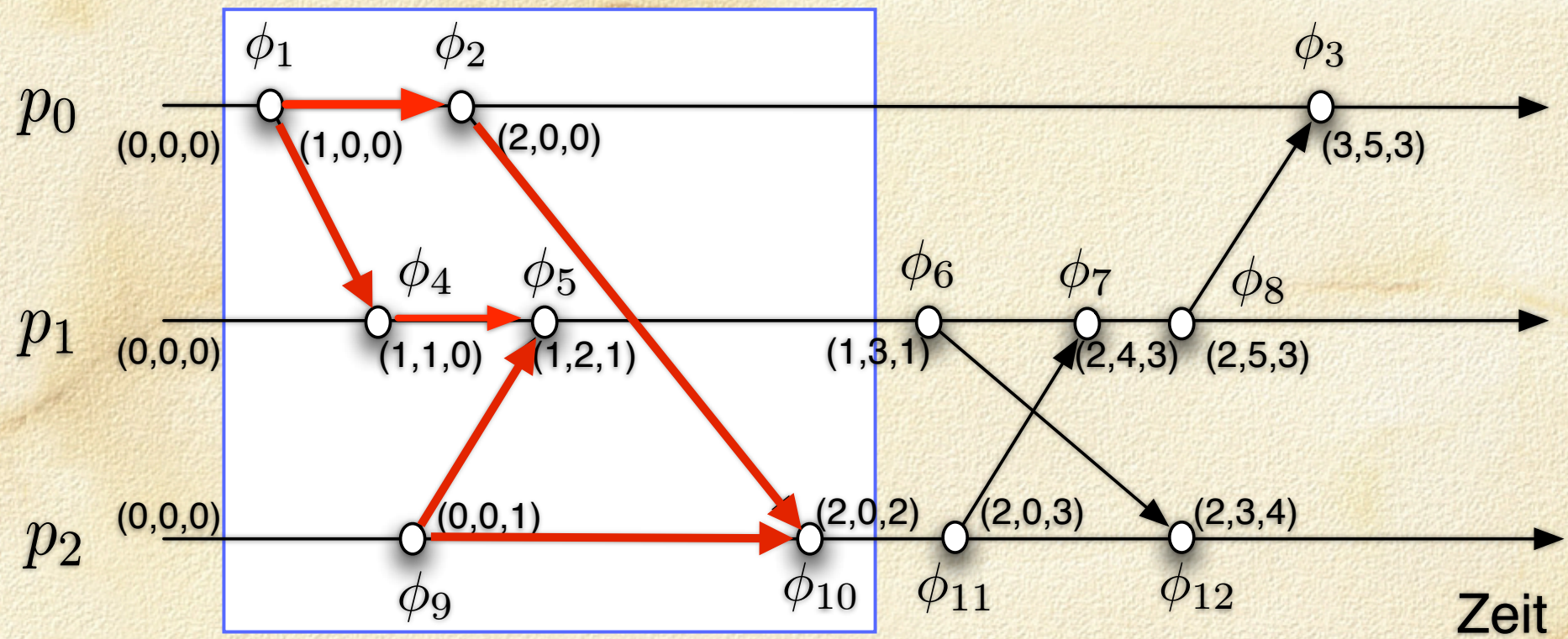
b) Strikte Ordnung *auf* $\mathbb{N}^n$ *:* $\vec{v}_1 < \vec{v}_2 :\Leftrightarrow \vec{v}_1 \leq \vec{v}_2 \wedge \vec{v}_1 \neq \vec{v}_2$

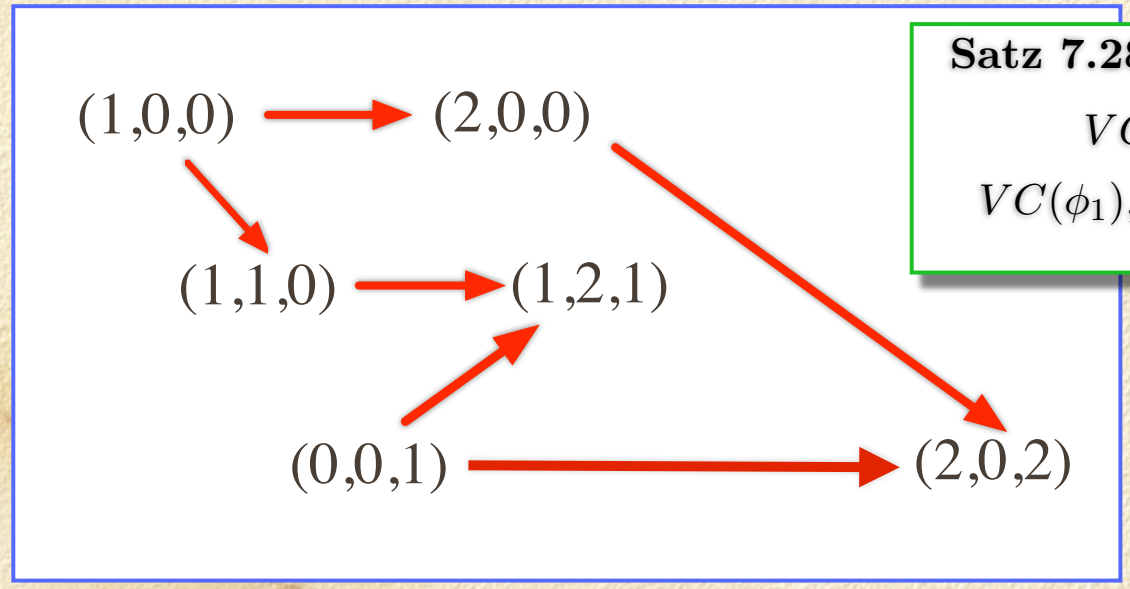
c) $\vec{v}, \vec{v}'$ *heißen* unvergleichbar, *falls* $\neg(\vec{v} \leq \vec{v}') \wedge \neg(\vec{v}' \leq \vec{v})$ *gilt.*



Definition 2.15 *ϕ_1 heißt* unabhängig *von ϕ_2 bzw. ϕ_1 heißt* nebenläufig *zu ϕ_2, geschrieben als $\phi_1 \parallel \phi_2$, falls gilt :*

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



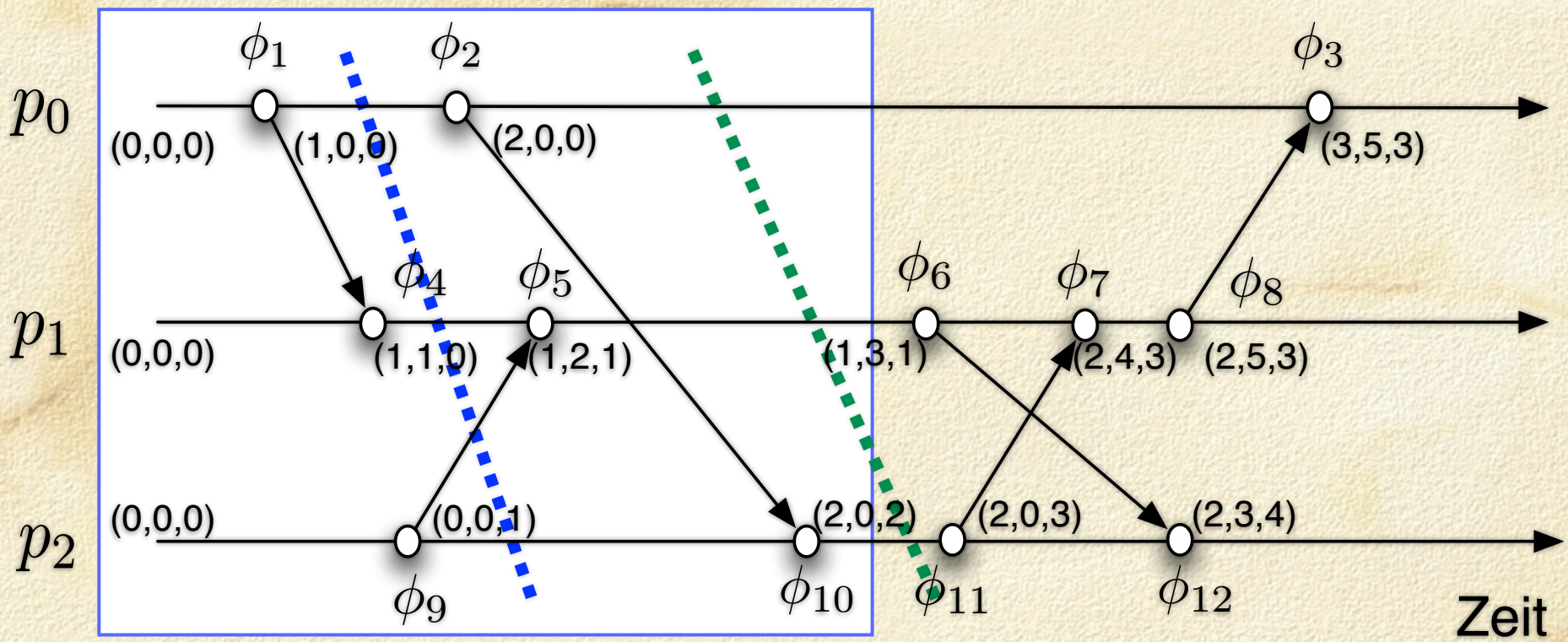


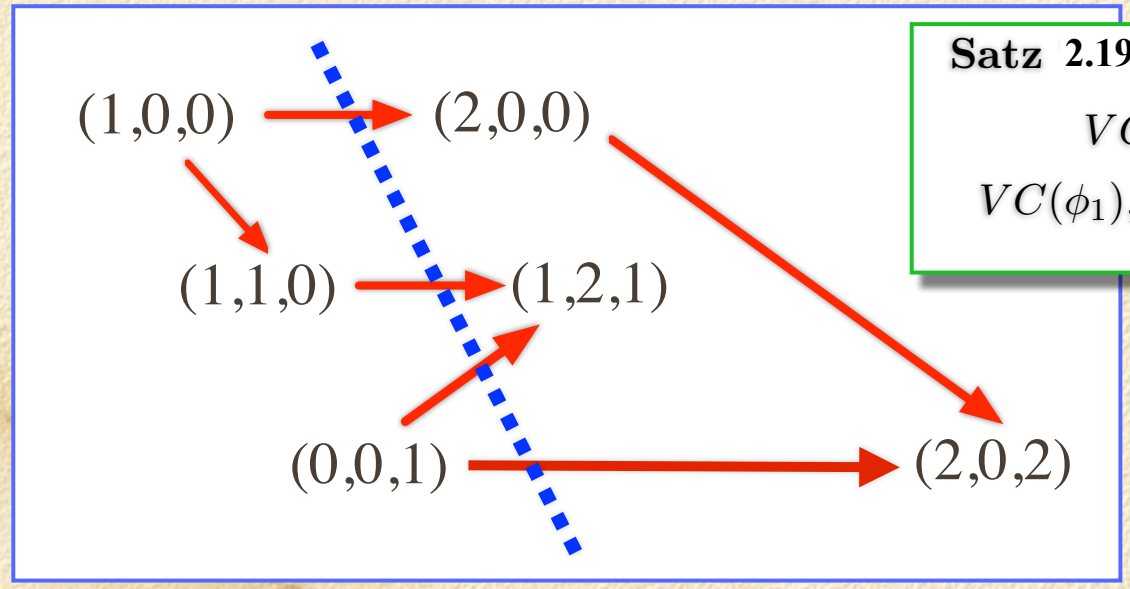
Satz 7.28

$$VC(\phi_1) < VC(\phi_2) \Leftrightarrow \phi_1 \; vor \; \phi_2$$

$$VC(\phi_1), VC(\phi_2) \; unvergleichbar \; \Leftrightarrow \phi_1 \parallel \phi_2$$

Aufgabe 2.20 Der Bankleitung werden ständig alle Vektor-Zeiten $VC(\phi)$ gesandt. Kann Sie darauf ein Verfahren aufbauen, um das Bilanz-Problem zu lösen?



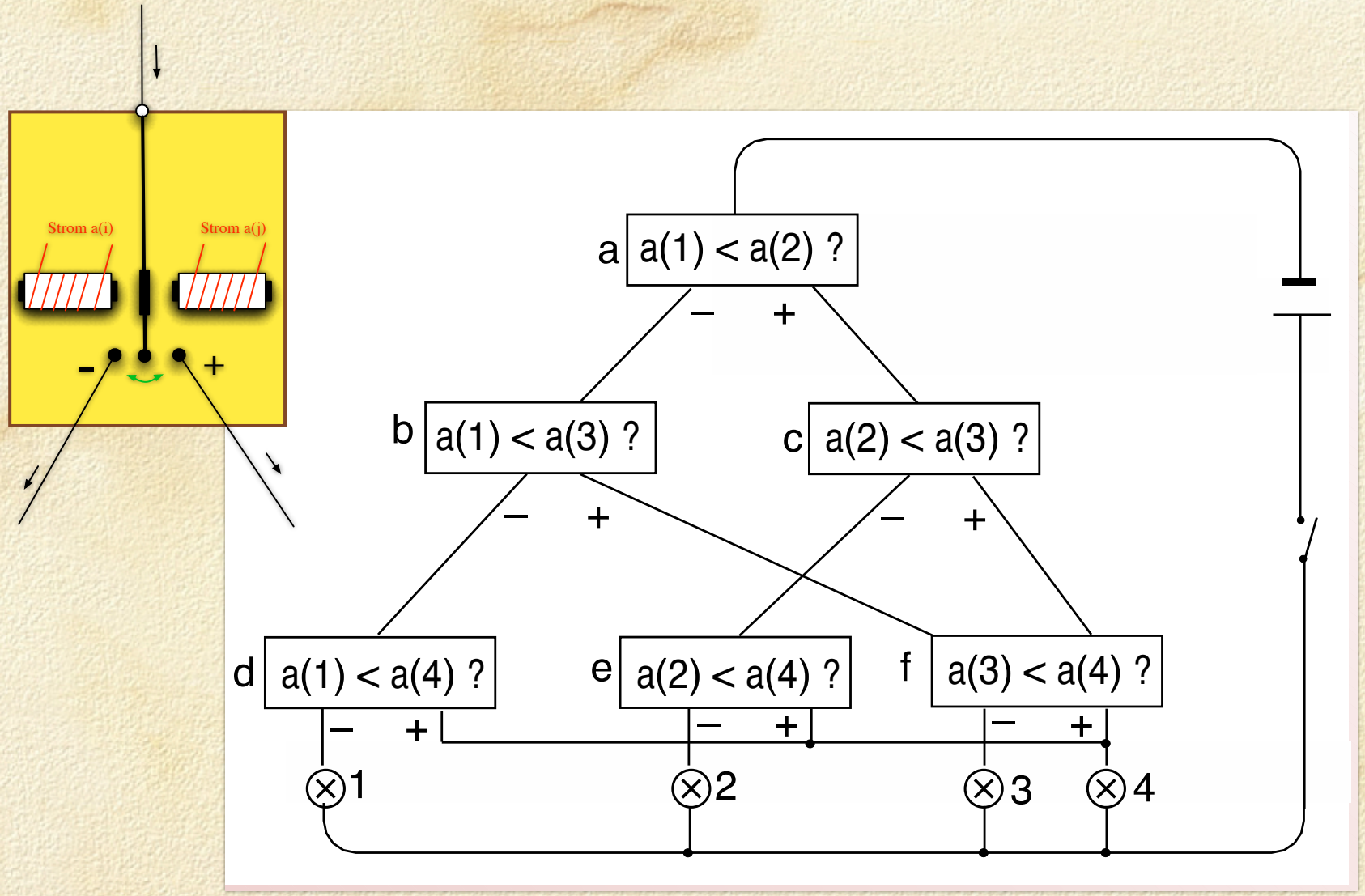


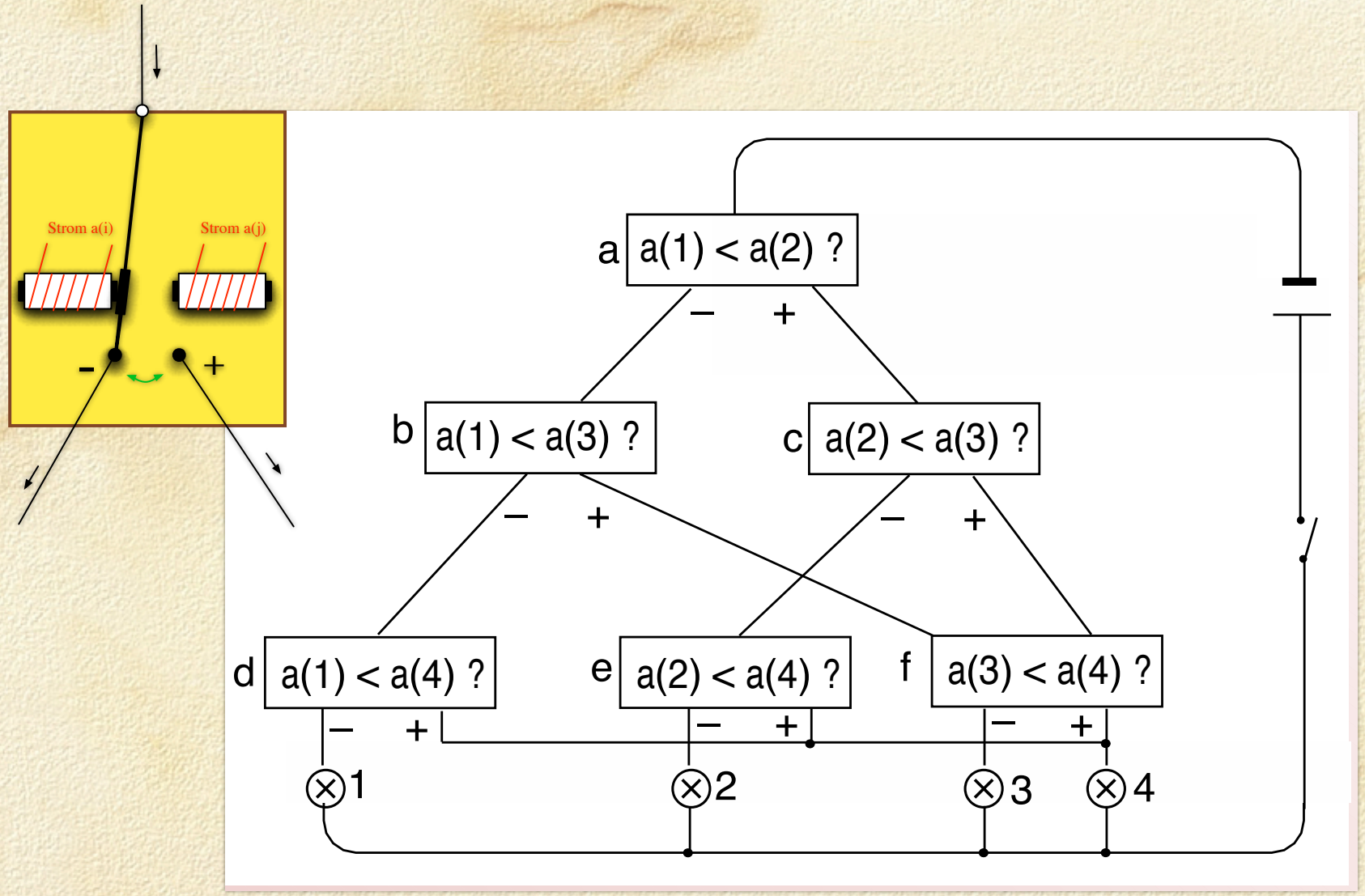
Satz 2.19

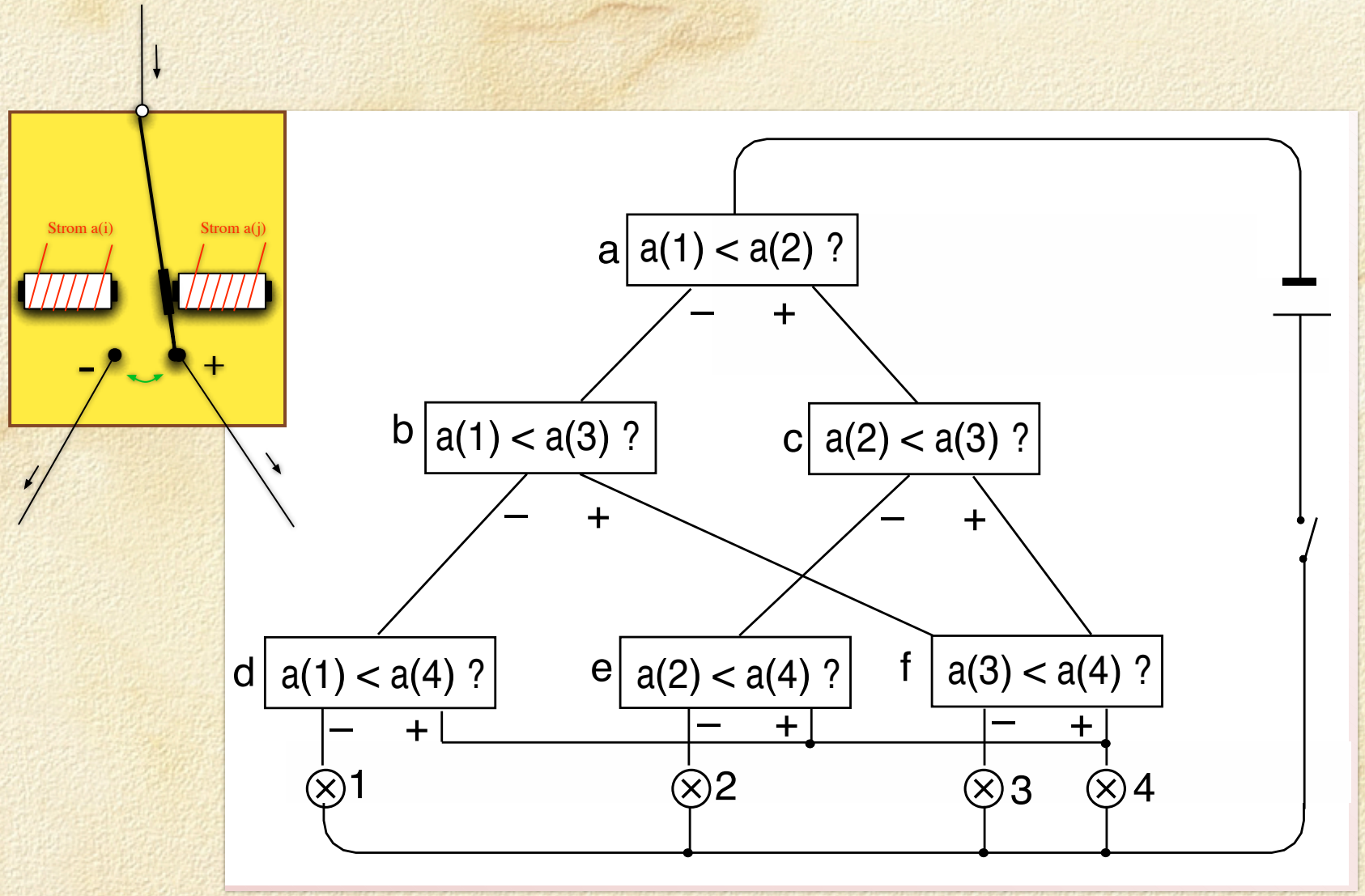
$$VC(\phi_1) < VC(\phi_2) \Leftrightarrow \phi_1 \; vor \; \phi_2$$

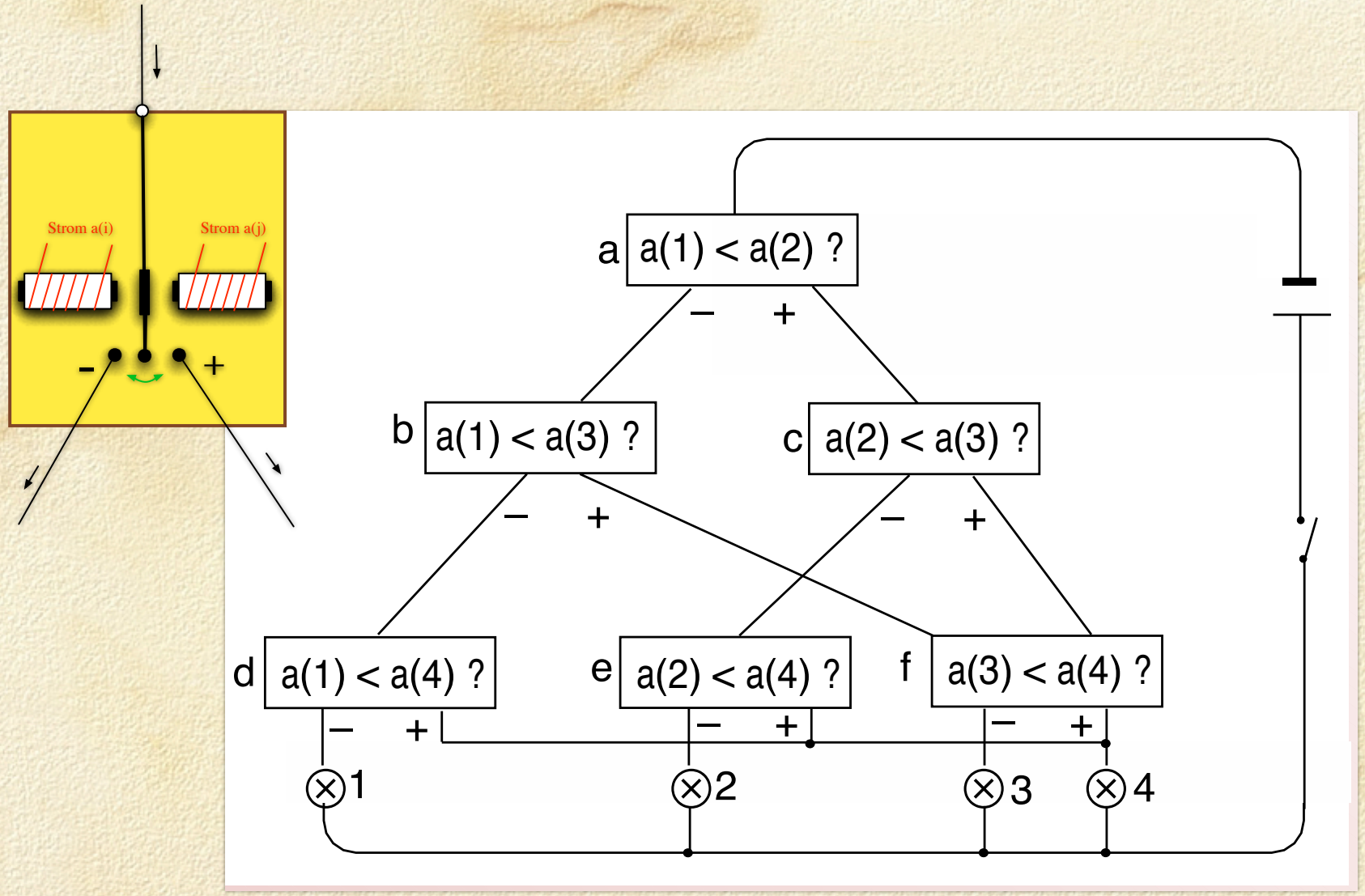
$$VC(\phi_1), VC(\phi_2) \; unvergleichbar \; \Leftrightarrow \phi_1 \parallel \phi_2$$

Wir erläutern den Begriff Nebenläufigkeit anhand eines **Beispiels von Dijkstra** : Gegeben seien vier paarweise verschiedene natürliche Zahlen $\{a(1), a(2), a(3), a(4)\} \subset \mathbb{N}$. Es soll eine Maschine konstruiert werden, die anzeigt, welche der vier Zahlen den größten Wert hat. Den Zahlen seien elektrische Ströme entsprechender Stärke zugeordnet, die die Relais paarweise ansteuern.


Strom a(i)
Strom a(j)
-
+
a
a(1) < a(2) ?
–
+
b
a(1) < a(3) ?
c
a(2) < a(3) ?
–
+
–
+
d
a(1) < a(4) ?
e
a(2) < a(4) ?
f
a(3) < a(4) ?
–
+
–
+
–
+
1
2
3
4


Strom a(i)
Strom a(j)
-
+
a
a(1) < a(2) ?
–
+
b
a(1) < a(3) ?
c
a(2) < a(3) ?
d
a(1) < a(4) ?
e
a(2) < a(4) ?
f
a(3) < a(4) ?
1
2
3
4

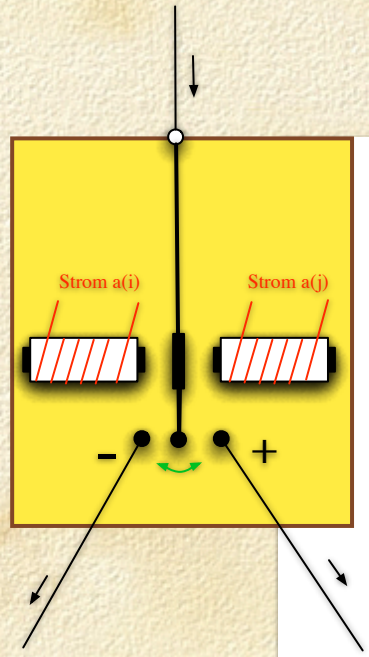

Strom a(i)
Strom a(j)
-
+
a
a(1) < a(2) ?
–
+
b
a(1) < a(3) ?
c
a(2) < a(3) ?
–
+
–
+
d
a(1) < a(4) ?
e
a(2) < a(4) ?
f
a(3) < a(4) ?
–
+
–
+
–
+
1
2
3
4

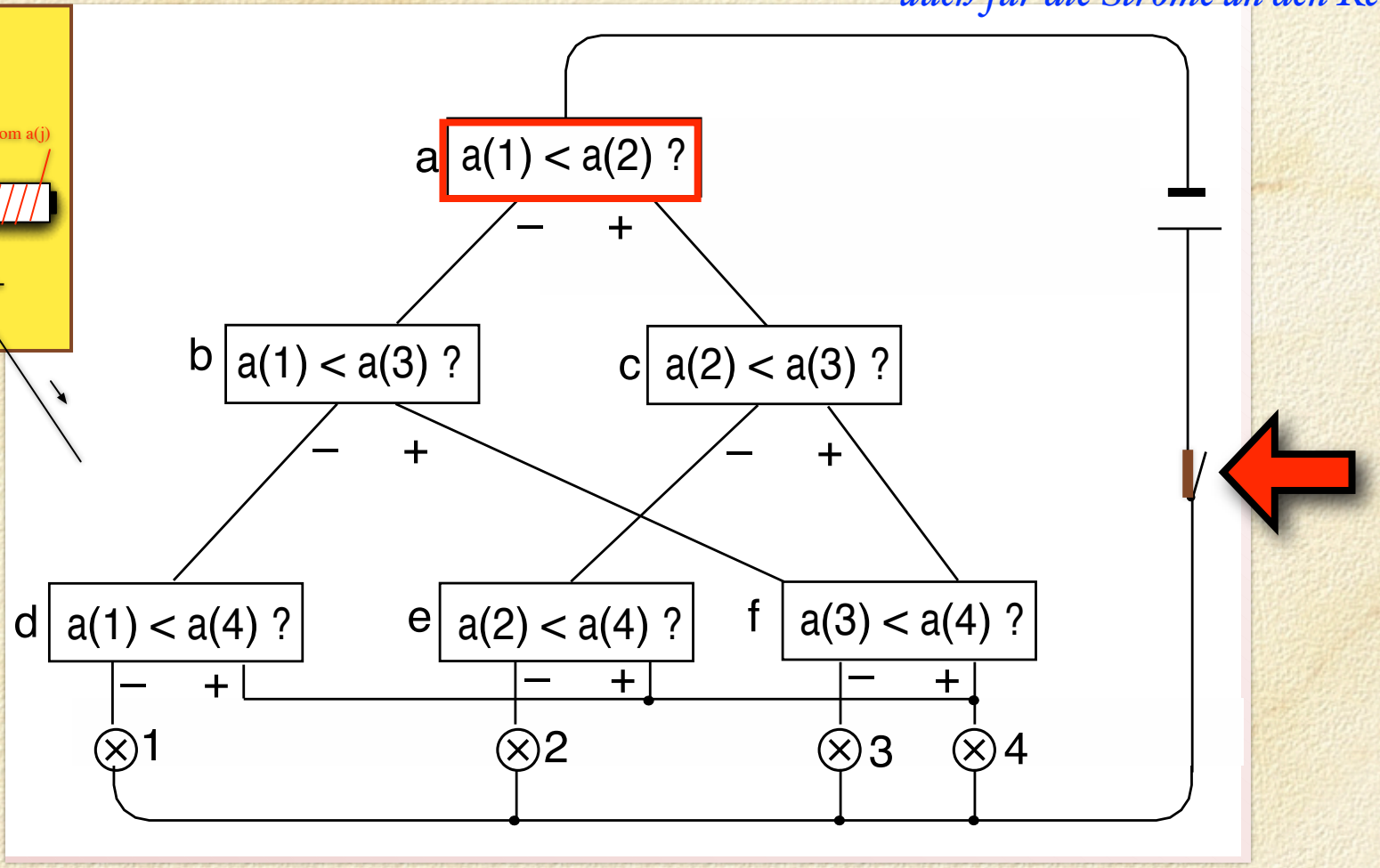


$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$

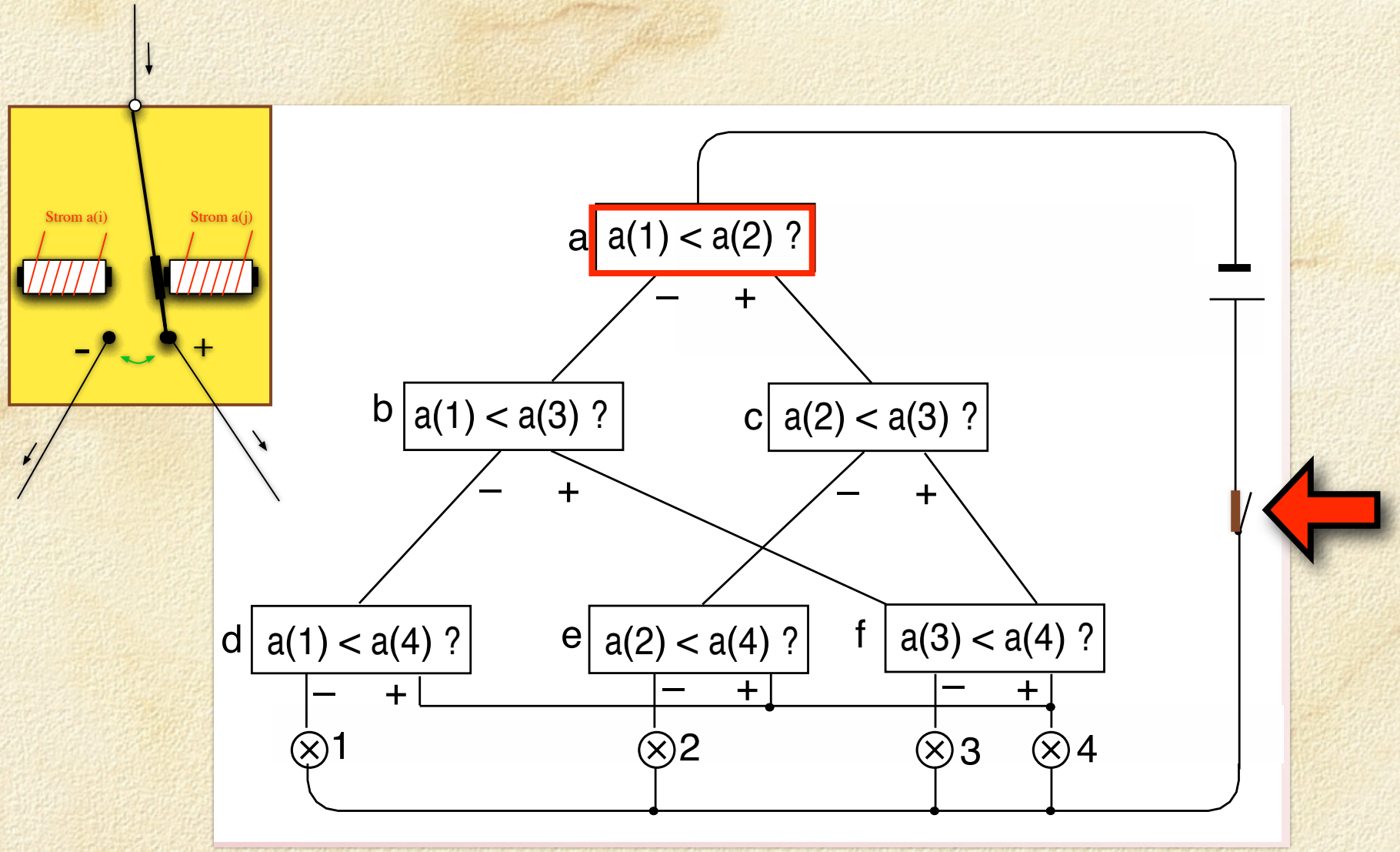
Was passiert, wenn der Schalter schließt?

auch für die Ströme an den Relais

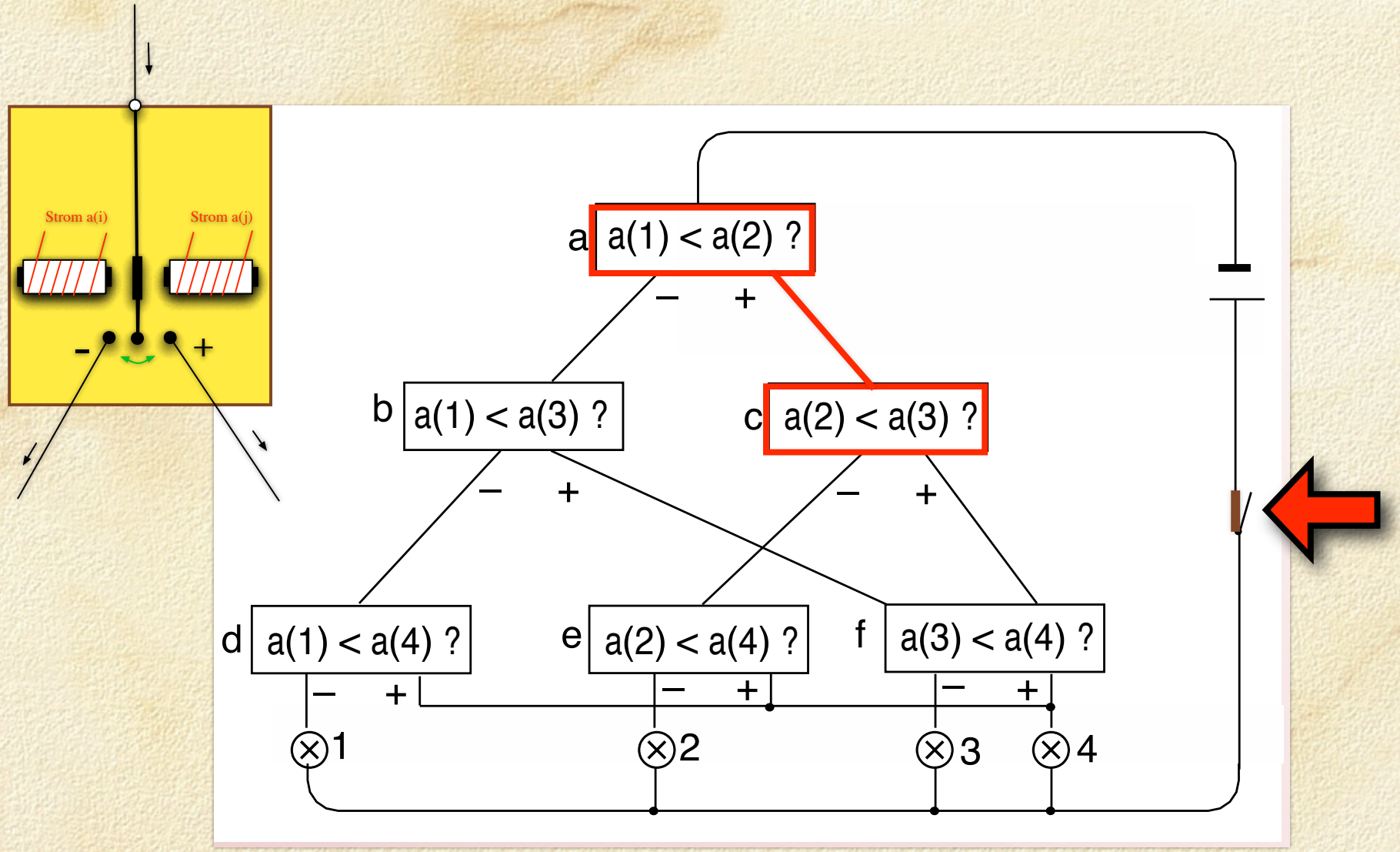




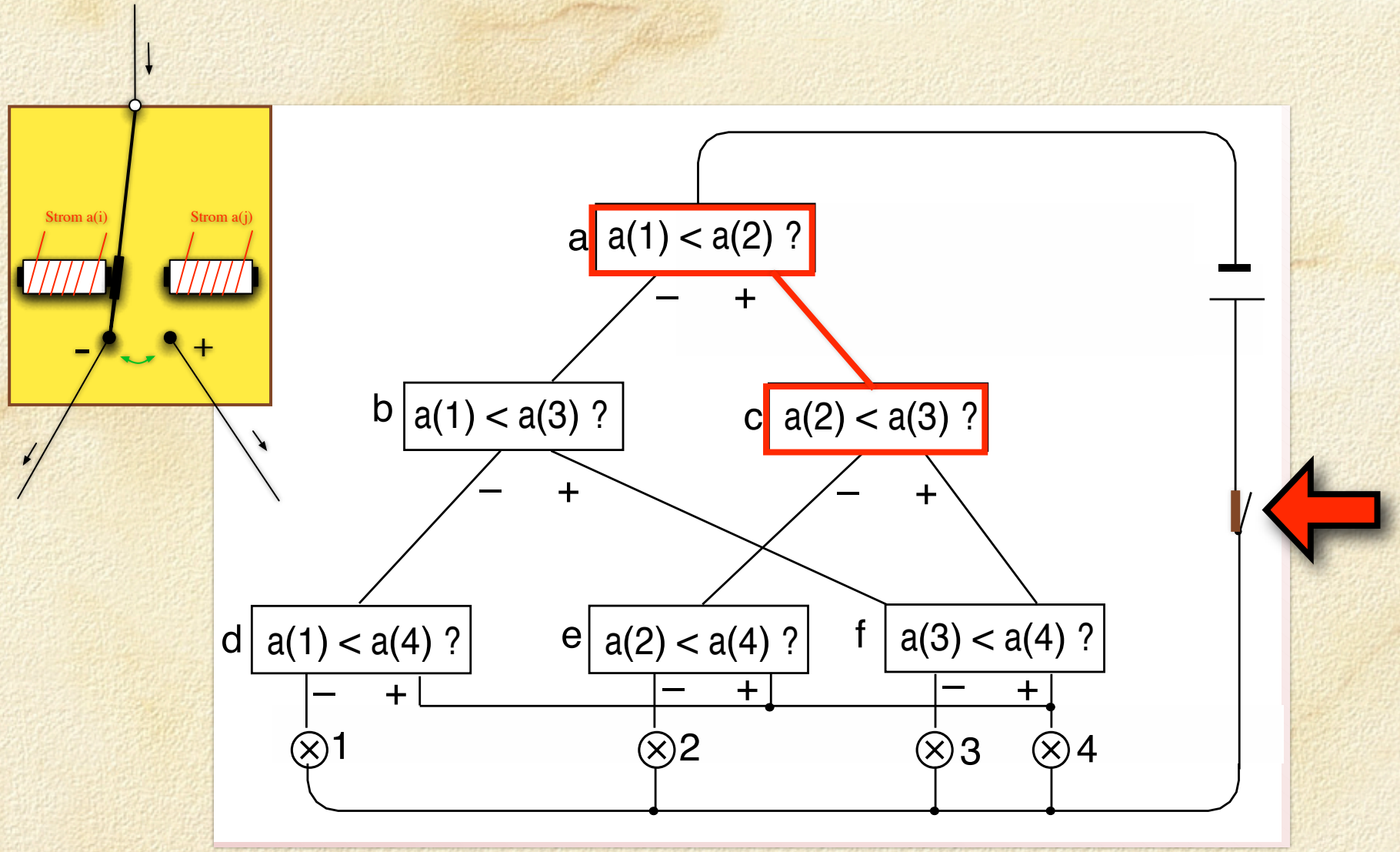
$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$



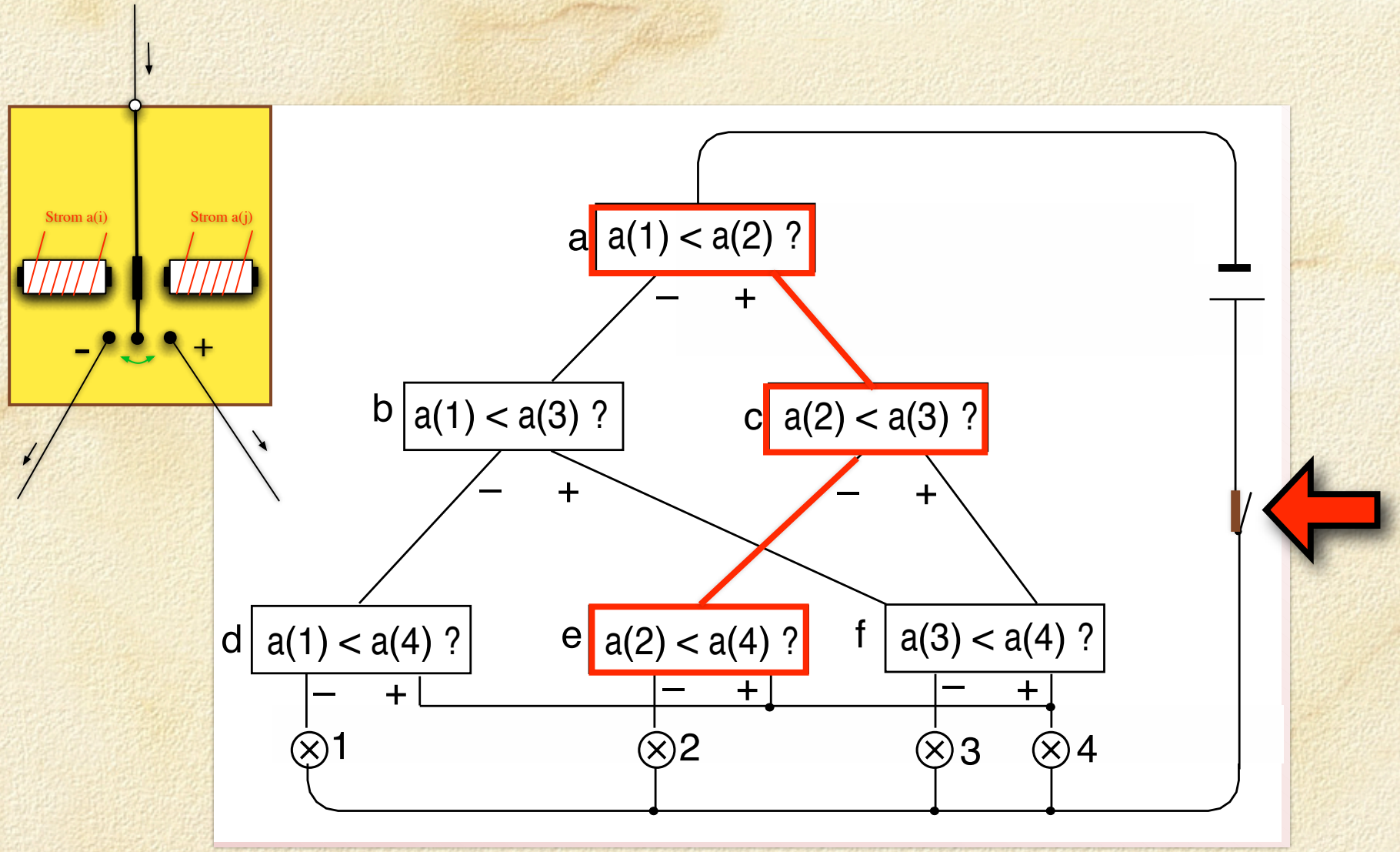
$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$



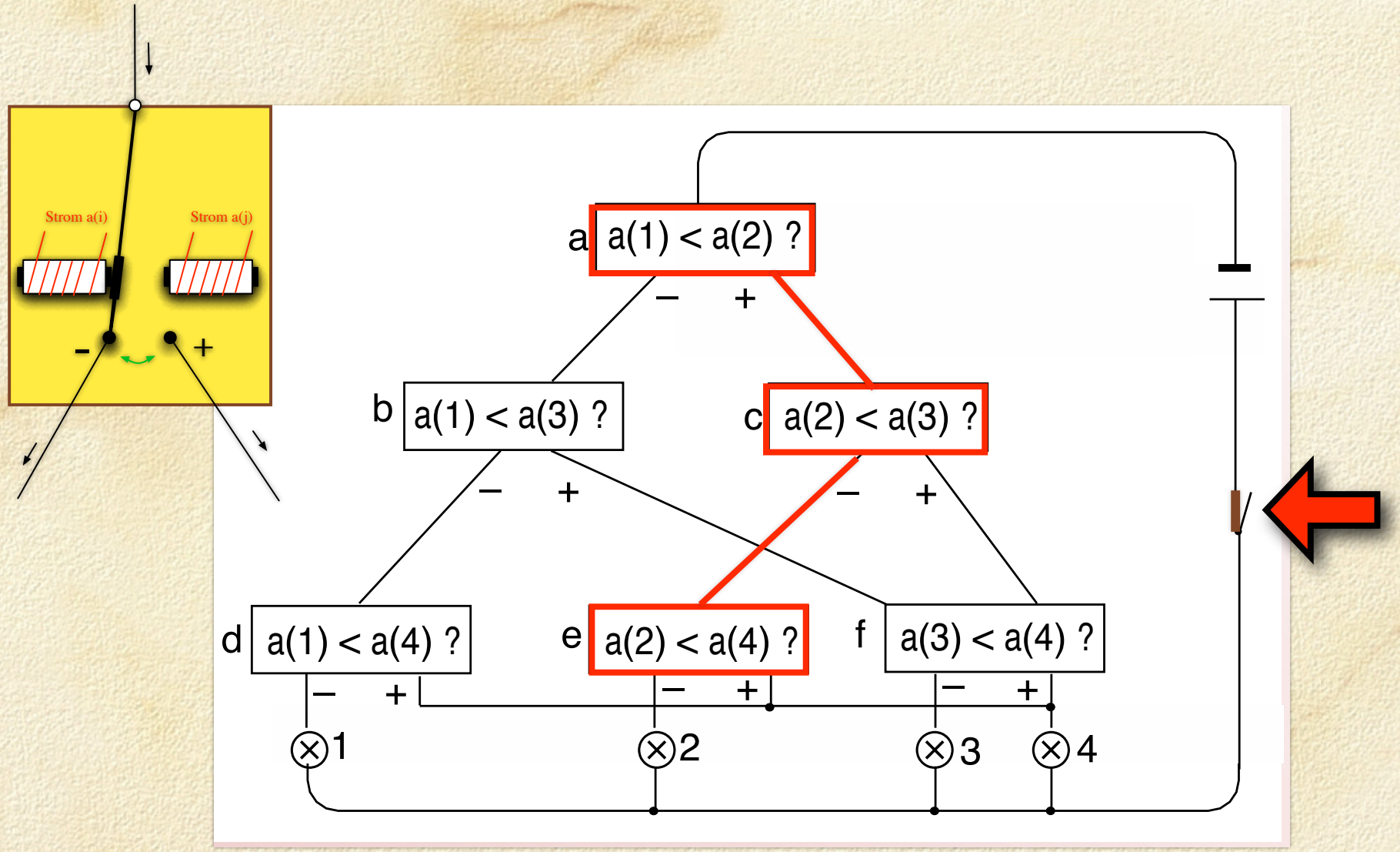
$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$



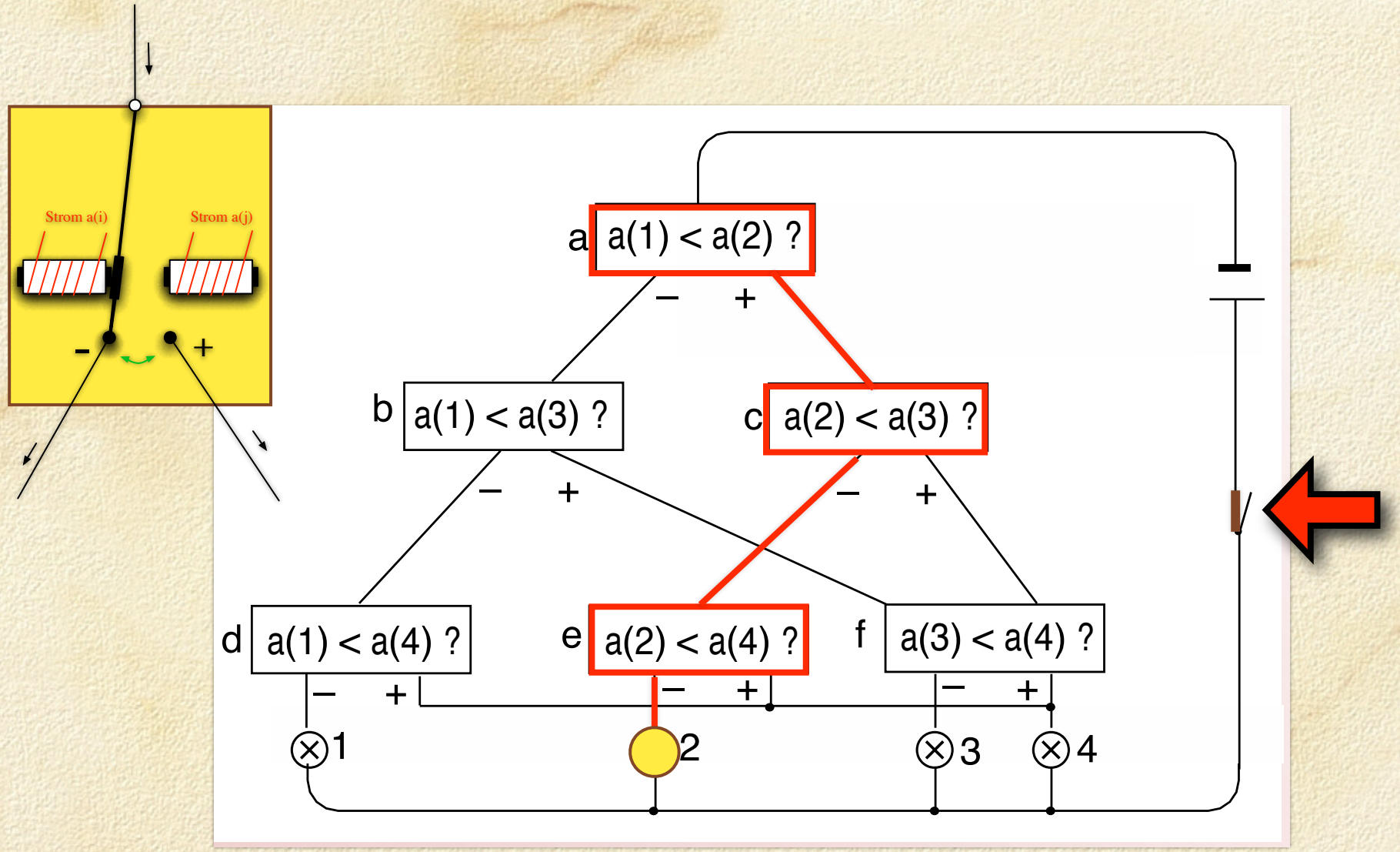
$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$



$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$

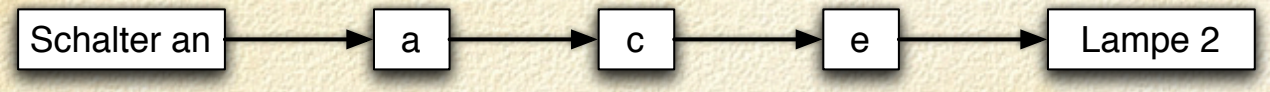


$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$

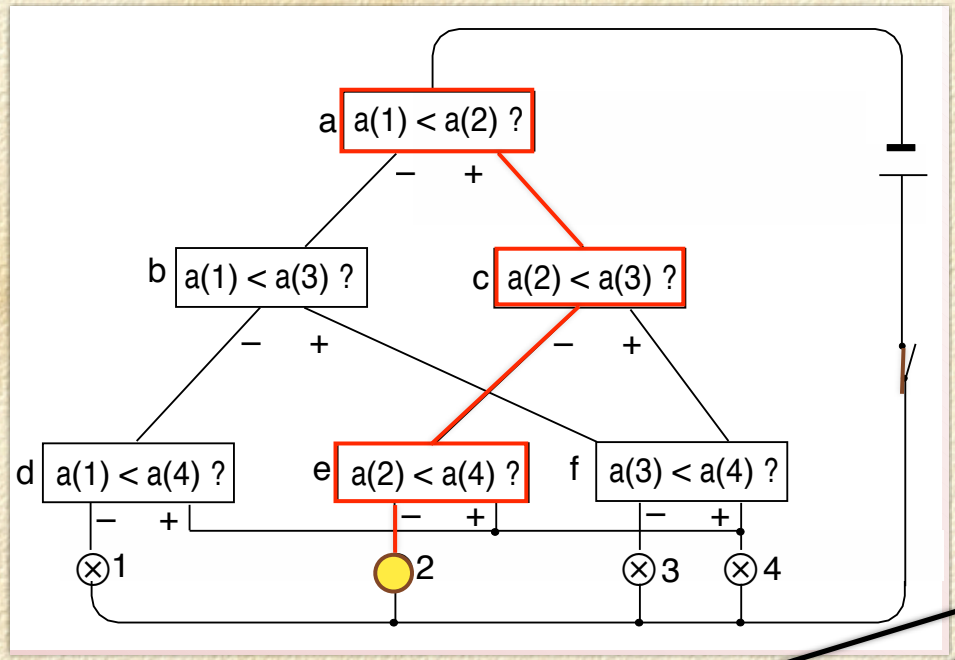


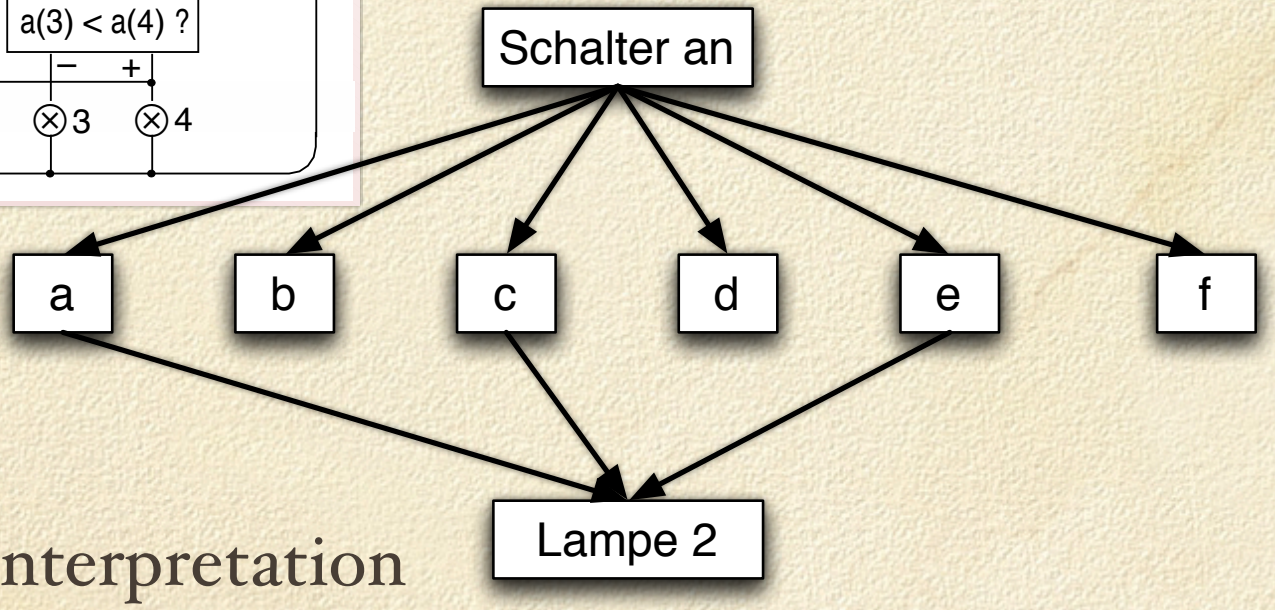
$$(a(1), \ldots, a(4)) = (7, 12, 2, 9)$$

Präzedenzen verschiedener Interpretationen



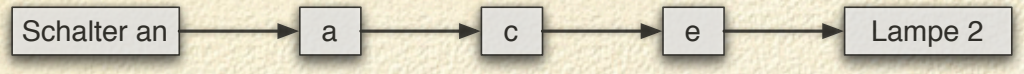
serielle Interpretation





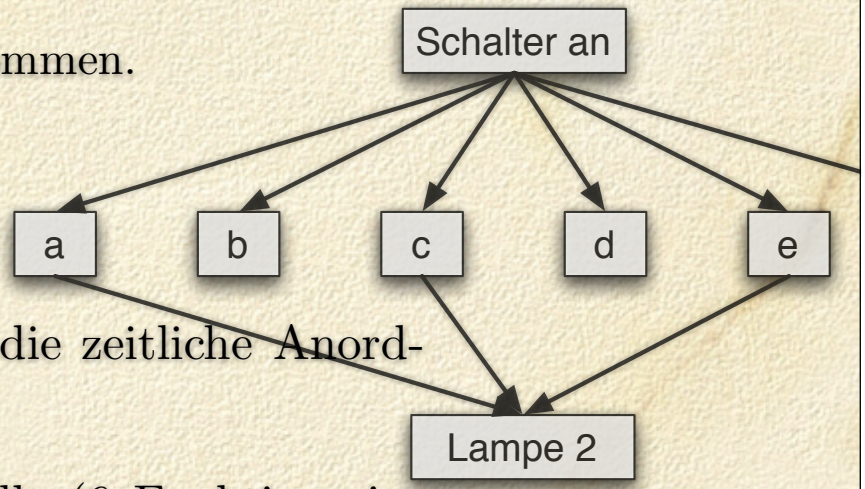
nebenläufige Interpretation

Bei der seriellen Interpretation wurde



- das Schaltbild der Relais-Anordnung (unnötigerweise) auf die zeitliche Anordnung übertragen,
- an nur eine ausführende Funktionseinheit gedacht (entsprechend der einzigen Linie und $co = id$),
- die dreifache Gesamtschaltzeit in Kauf genommen.

Dagegen wird bei der nebenläufigen Interpretation



- nur die tatsächliche kausale Abhängigkeit in die zeitliche Anordnung übernommen,
- die größte mögliche Nebenläufigkeit dargestellt (6 Funktionseinheiten entsprechend der Elementezahl des größten Schnitts).
- die Gesamtschaltzeit nur durch die maximale Relaisschaltzeit bestimmt.