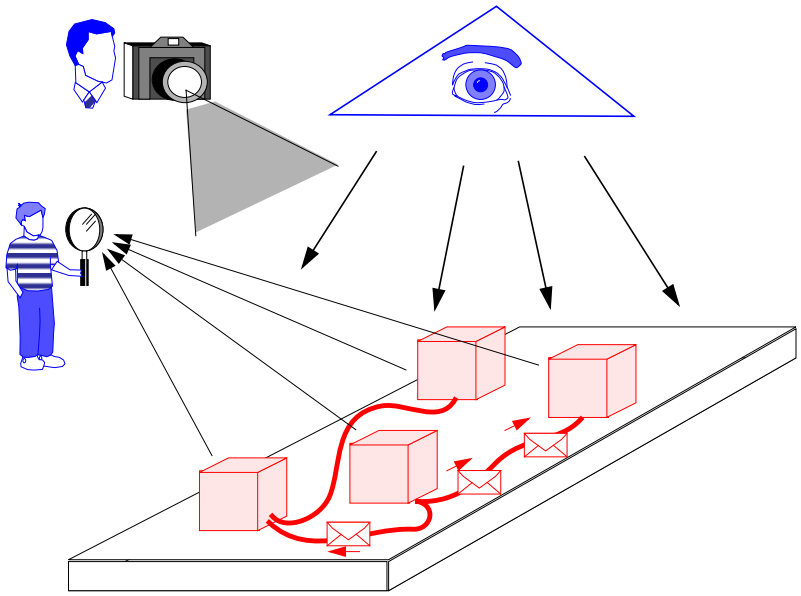
Globale Zustände, Beobachtungen, Prädikate



Das Schnappschussproblem

Problem: "Momentaner" *Schnappschuss* des globalen Zustands, ohne das System anzuhalten

Realität:

- *Volkszählung*: Stichzeitpunkt (geht hier nicht)
- *Inventur*: Einfrieren (unpraktisch)

Anwendungen:

- konsistenter Aufsetzpunkt für vert. Datenbanken
- wie hoch ist die momentane Last?
- Testen verteilter Systeme (gilt eine globale Eigenschaft?).
- Deadlock: Existiert eine zykl. Wartebedingung?
- ist die verteilte Berechnung terminiert?
- ist ein bestimmtes Objekt "Garbage"?
- ...

} *Prädikate* über glob. Zuständen

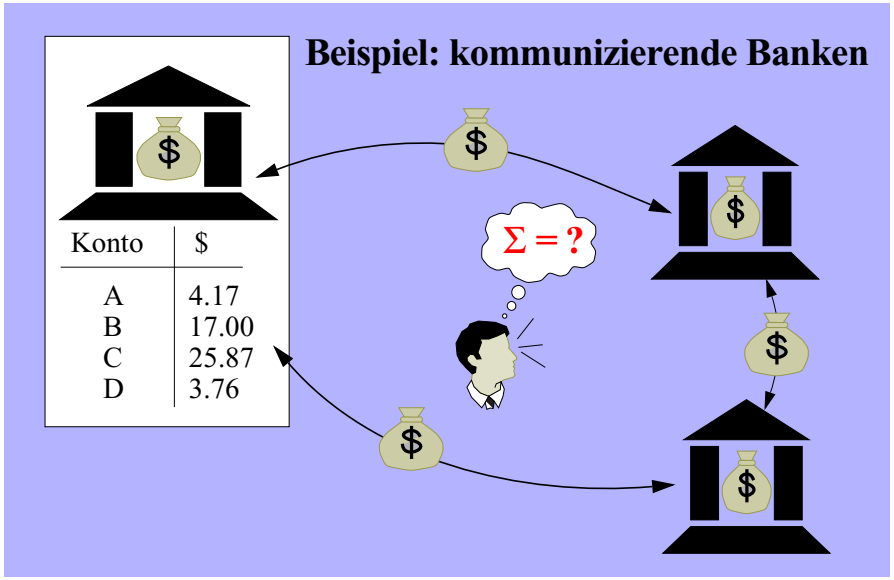
Schwierigkeiten:

- unmöglich, alle Prozesse gleichzeitig zu erwischen
- unbestimmte Nachrichtenlaufzeiten
- Nachrichten, die unterwegs sind, sieht man nicht
- ermittelter Zustand ist i.a. veraltet
- ... u.U. nie "wirklich" so gewesen
- ... u.U. inkonsistent

zumindest dies ausschliessen!

→ *Schnappschussalgorithmus*

Beispiel: Kommunizierende Banken



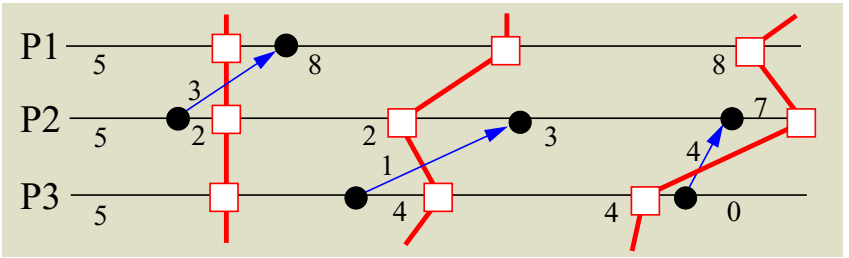
- Modellierung:
  - ständige Transfers zwischen den Konten bzw. Banken
  - lokale atomare Aktionen: alle Geldkonten *einer* Bank können "gleichzeitig" (also atomar und damit "lokal konsistent") untersucht werden

- Wieviel Geld ist in Umlauf?
  - konstante Geldmenge, oder
  - monotone Inflation (→ untere Schranke für momentane Geldmenge)

- Erschwerte Bedingungen:
  - niemand hat eine globale Sicht
  - gemeinsame Zeit?

(In)konsistente Schnitte

Beispiel: Wieviel Geld ist in Umlauf?



idealer (senkrechter) Schnitt → 15

Unerreichbar

konsistenter Schnitt → 15

Äquivalent zu senkrechtem Schnitt (lässt sich problemlos "geradebiegen")

inkonsistenter Schnitt → 19 (+4 ?)

Lässt sich nicht "geradebiegen"

manches doppelt (bzw. gar nicht) gezählt

"Gummiband-transformation"

→ verändert Metrik

→ erhält Topologie

Wie wir noch einsehen werden:

Kausaltreues Beobachten ⇔ konsistente Schnitte
- wie erreicht man das?

Schnappschussalgorithmen: Zweck

auch wenn der Zustand nur möglich gewesen *wäre*, aber gar nicht *wirklich* auftrat!

aber was heisst schon "wirklich"?

- Liefern *konsistenten, möglichen, vergangenen* Zustand

  - *konsistenter Zustand* = globaler Zustand entlang einer konsistenten Schnittlinie (keine Nachricht "aus der Zukunft")

  - nur über konsistenten Zuständen können *globale Prädikate* sinnvoll bestimmt werden (da diese "äquivalent" zu Zuständen entlang senkrechter Schnittlinien sind); in diesem Sinne sind solche Algorithmen wichtig!

  - ein Prädikat heisst *stabil* (oder monoton), wenn es nie wieder aufhört zu gelten, nachdem es gilt (also in allen zukünftigen Zuständen gilt); z.B. Terminierung, Garbage, Deadlock...

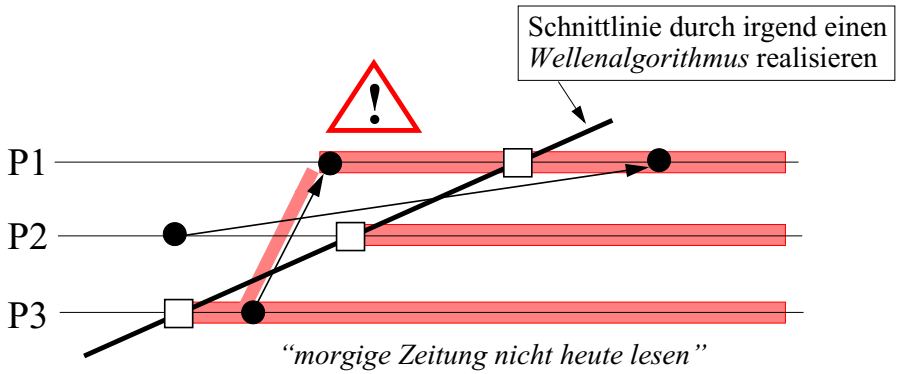
- Falls Prädikat stabil: "entdecken" dieses Prädikat (⇒ "stable property detection algorithm")

  - bei stabilen Prädikaten ist "möglich... vergangen" sogar brauchbar! (es gilt dann jetzt sicherlich)

  - aber wenn die betrachtete Eigenschaft nicht stabil ist, was dann?

  - wer garantiert eigentlich, dass eine Eigenschaft (wirklich) stabil ist?

Ein Schnappschussalgorithmus



Prozesse, Nachrichten: *schwarz* oder *rot*

Schnappschussaugenblick: schwarz → rot
(dann: lokalen Zustand dem Initiator melden)

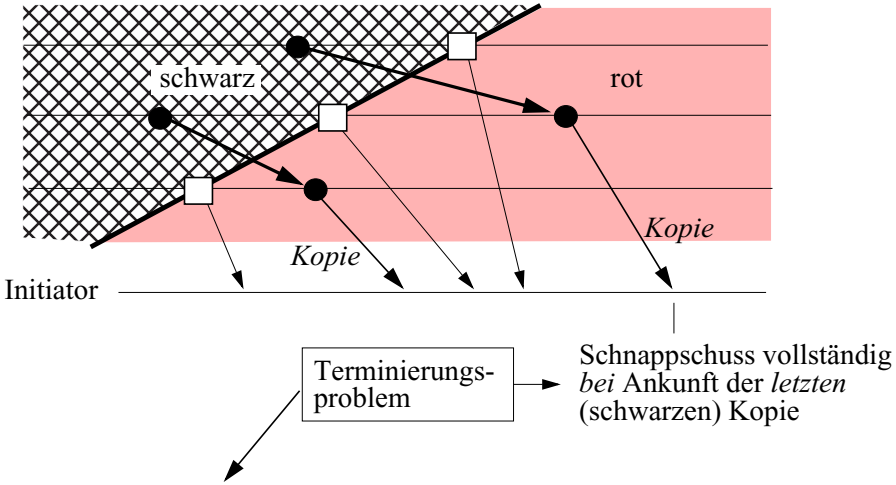
Prozess wird rot, wenn a) Aufforderung erhalten,
b) Erhalt einer roten Nachricht

Beh.: Schnappschuss ist *konsistent*.
Bew.: Keine "Nachricht aus der Zukunft"

Nachrichten, die *unterwegs* sind?
- Schwarze Nachrichten, die bei rot ankommen
- Sende *Kopie* davon an Initiator
- Problem: Wann letzte Kopie dort eingetroffen?

Schnappschussalgorithmus: Nachrichten

- In-Transit-Nachrichten?
  - schwarze Nachrichten, die von einem roten Prozess empfangen werden
  - Sende (bei Empfang) eine Kopie davon an den Initiator
  - Problem: Wann hat der Initiator die letzte Kopie erhalten?



- Z.B. "Defizitzähler" als Teil des konsistenten Zustands
  - zähle gesendete und empfangene schwarze Nachrichten
  - globale Differenz = Anzahl zu erwartender schwarzer Kopien
- Wellenalgorithmus als "*Basisalgorithmen*" → unterschiedliche Schnappschussalgorithmen

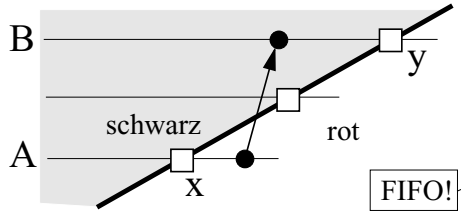
---

- FIFO-Kommunikation keine Voraussetzung
- Letzte In-transit-Nachricht abwarten
  - das kann aber lange dauern; geht es nicht auch schneller? (ja! wie?)
- "Repeated snapshot": Farben vertauschen

Der Chandy/Lamport-Algorithmus

(erster Schnappschussalgorithmus, 1985 veröffentlicht in ACM TOCS 3, 63-75)

- *Idee:* 1) Setzt FIFO-Kanäle voraus ("flushing-Prinzip")
  Marker schieben In-transit-Nachrichten aus den Kanälen heraus
  2) Flooding als zugrundeliegendes Wellenverfahren
- ¬ ∃ Nachricht aus der Zukunft ⇒ Schnitt ist *konsistent*
  - vgl. auch frühere Ausführungen zu "virtuell gleichzeitiges Markieren"



- bei x wird ein *Marker* von A nach B gesendet (falls A und B benachbart sind)
- Nachricht kann den Marker nicht überholen ⇒ nach x gesendete Nachricht kommt *nach* y an

---

- *In-transit-Nachrichten* bei FIFO-Kanälen:
  - Nach der letzten schwarzen Nachricht folgt ein Marker
  - Empfang eines Markers informiert den Empfänger, dass nun über diesen Kanal keine schwarzen Nachrichten mehr ankommen

---

- *Vorteil*: Farben müssen nicht (in Nachrichten) mitgeführt werden
- *Nachteile*:
  - bei dichten Netzen grosse Zahl von Kontrollnachrichten
  - FIFO ist notwendig
  - lokale Zustände müssen i.a. zum Initiator gebracht werden (z.B. mittels Echo-Nachrichten)

Der Chandy/Lamport-Algorithmus (2)

- Globaler Zustand besteht aus den *Prozesszuständen* und allen *Kanalzuständen*

- Im Unterschied zum nicht-FIFO-Fall sind Kanalzustände *Folgen* von Nachrichten, keine Mengen

  Channel State = sequence of messages sent along the channel before the sender's state is recorded, excluding the sequence of messages received along the channel before the receiver's state is recorded

Marker-Sending Rule for a Process p. For each channel *c*, incident on, and directed away from *p*:

p sends one marker along *c* after *p* records its state and before *p* sends further messages along *c*

Marker-Receiving Rule for a Process q. On receiving a marker along channel *c*:

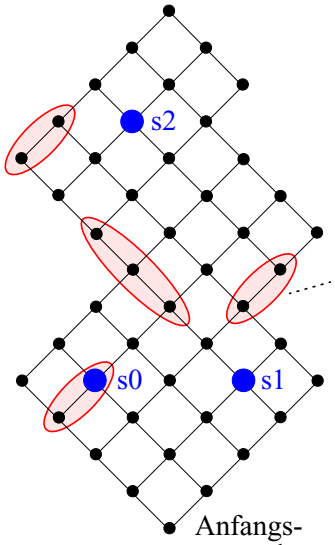
if *q* has not recorded its state **then**
q records its state;
q records the state of *c* as the empty sequence
else
q records the state of *c* as the sequence of messages received along *c* after *q*'s state was recorded and before *q* received the marker along *c*

- Wie verhält sich ein Initiator? Kann der Algorithmus "spontan" von mehreren Prozessen unabhängig voneinander initiiert werden?

- Wieso ist im ersten Fall der Kanalzustand die leere Folge? Hat jeder so gewonnene globale Zustand einige leere Kanäle?

Eigenschaften verteilter Berechnungen mit Schnappschüssen "entdecken"?

- Ein wiederholt angewendeter Schnappschussalgorithmus könnte zuerst s1, dann s2 liefern (s2 ist "später" als s1)
  - dazwischen sind Lücken!



Menge der konsistenten Zustände der Berechnung, geordnet entsprechend der "später-Relation" (entspricht der Präfixbeziehung)

Eigenschaft sei in diesen Bereichen gültig (wird jedoch von s1 oder s2 nicht "entdeckt", allenfalls von s0)

wenn dies in einem Zustand gilt, *dann* auch in allen späteren

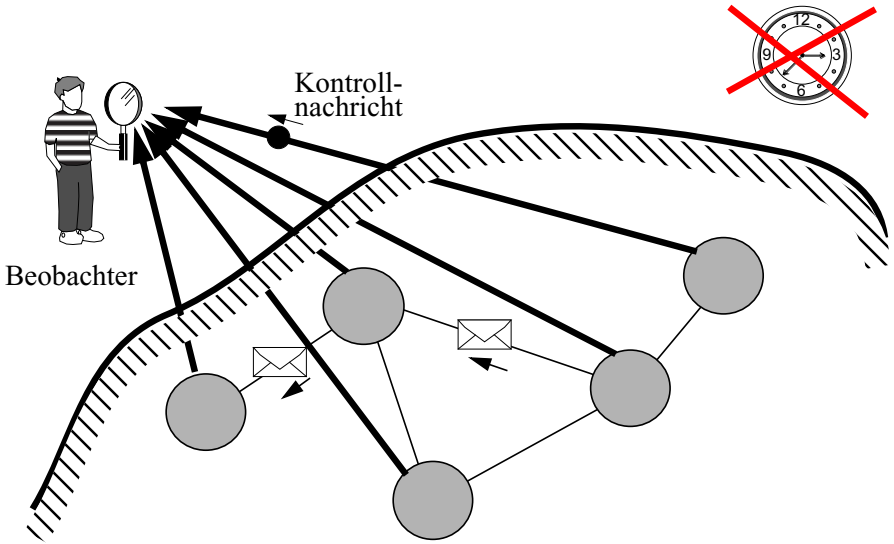
- Sinnvoll, wenn die Eigenschaft stabil ist - aber ansonsten?
  - beachte: wir wissen nicht, ob "in Wirklichkeit" s0 oder s1 eintrat!

- Wir hätten gerne eine lückenlose "Folge" konsistenter Schnappschüsse als eine "*Beobachtung*" der Berechnung

- Allerdings sind Berechnungen nur *halbgeordnete* Mengen (konsistenter) Zustände (also keine Folgen)!

Beobachten verteilter Berechnungen

wie früher schon erwähnt...:

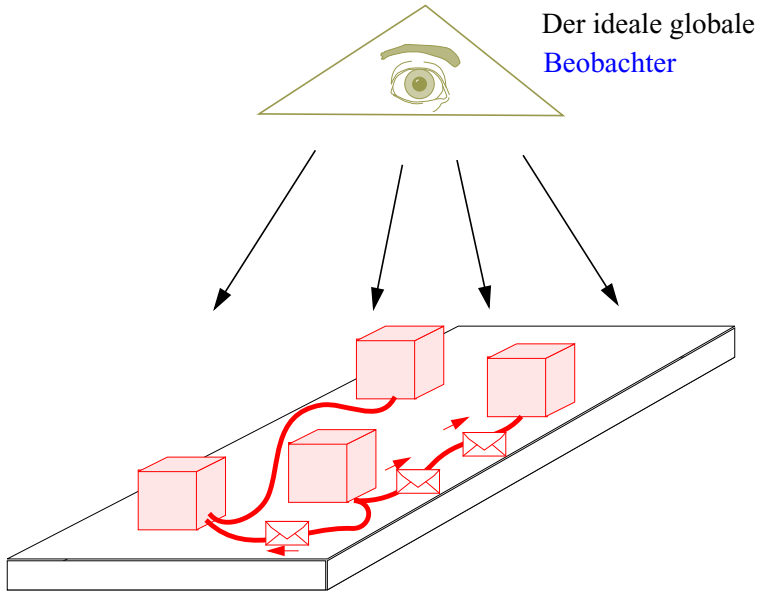


Beobachten geht nur über das Empfangen von "Kontrollnachrichten" (mit unbestimmter Laufzeit)

"Axiom": Mehrere Prozesse können "niemals" gleichzeitig beobachtet werden

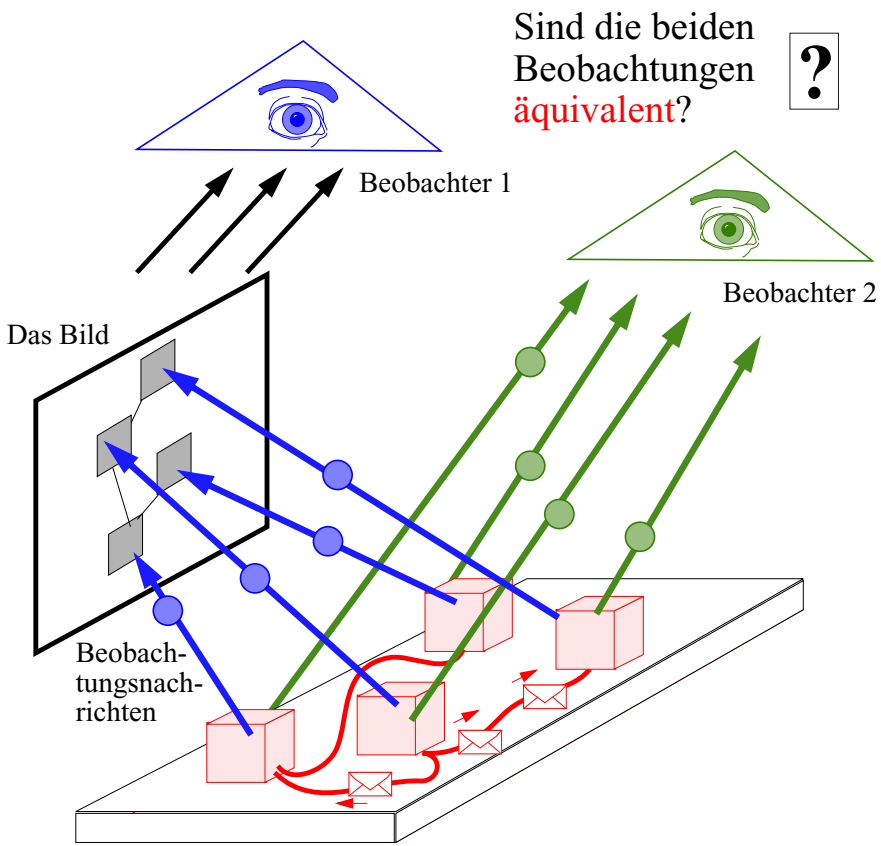
"Korollar": Aussagen über den globalen Zustand sind schwierig

Verteilte Berechnung und Beobachtung



Eine verteilte Berechnung

Beobachtungen...



Probleme:

- Zeitverzögerung der Beobachtung
- Konsistenz des Bildes
- Verzerrung des Verhaltens ("Heisenberg'sche Unschärfe")

Das Paradigma der Beobachtung

- "Korrektes" Beobachten verteilter Systeme ist ein wichtiges praktisches Problem

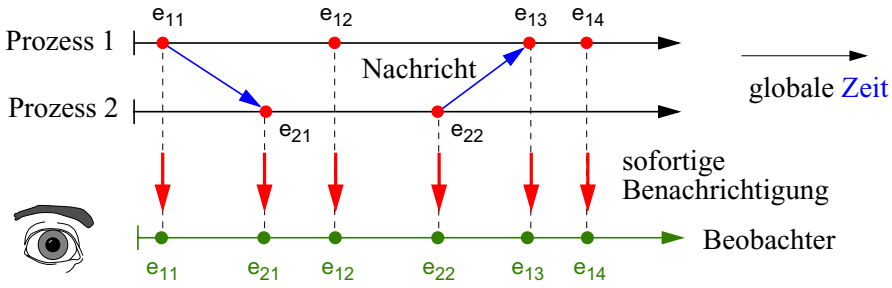
- Kausaltreue (auch: "kausal konsistente") Beobachtungen stellen das Kernproblem vieler verteilter Algorithmen dar

---

⇒ Wie realisiert man kausaltreue Beobachter?

⇒ Was sind, formal gesehen, *kausaltreue Beobachter / Beobachtungen ?*

Ideale Beobachtungen



wesentliche Vorkommnisse, z.B.
Nachricht empfangen / senden

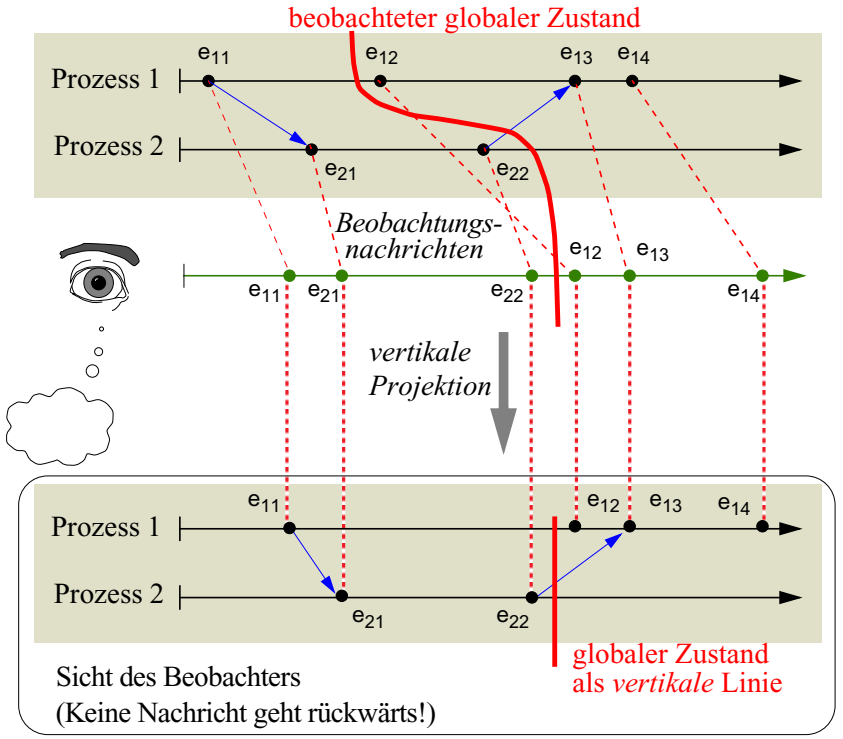
- Beobachtet werden Ereignisse bei den Prozessen
  - Meldung nach aussen über Beobachtungsnachrichten

- Eine solche ideale Beobachtung möchten wir haben, können das aber nicht garantiert bekommen

- Statt dessen könnten wir etwas bekommen, was wir nicht haben wollen...
  (nämlich eine inkonsistente Beobachtung, bei der sich Beobachtungs-nachrichten so überholen, dass Ursache und Wirkung vertauscht werden)

Kausaltreue Beobachtungen

- Ursachen werden stets vor ihren Wirkungen angezeigt
  - dies hoffen wir zu bekommen! (aber wie?)



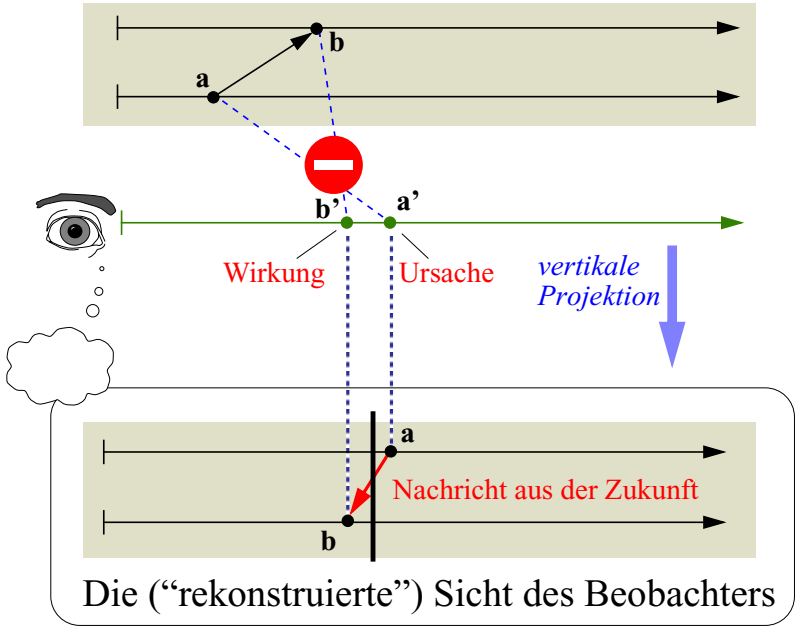
- Beobachtersicht ist nur in irrelevanter Weise verzerrt
  (d.h. ist eine Gummibandtransformation des echten Ablaufes)

- Beobachteter globaler Zustand ist daher *möglich*

  - *gültige*, nicht zu widerlegende Interpretation
  - nicht entscheidbar, ob *tatsächlich eingetreten*

Kausal inkonsistente Beobachtungen

- Überholungen von Benachrichtigungen:



- In der Interpretation des Beobachters:
  - Nachricht fliesst rückwärts in der Zeit
  - Kausalität ist verletzt (Wirkung vor Ursache!)
  - Beobachteter globaler Zustand nach b ist inkonsistent

- Wir hätten gerne eine kausaltreue Beobachtung, wo die Beobachtersicht nur in irrelevanter Weise verzerrt ist
  - d.h. eine Gummibandtransformation des echten Ablaufes ist (bei der keine Nachricht rückwärts verläuft)
  - wie erzwingt man das?

Kausaltreue Beobachtungen

lineare Erweiterung oder Einbettung

Definition:
Eine *kausaltreue Beobachtung* einer Berechnung ist eine Linearisierung der entsprechenden (partiellen) Kausalordnung (E,<)

Mit anderen Worten:
Jeder "Trace" von Ereignissen, in dem eine Wirkung niemals vor ihrer Ursache erscheint, heisst "kausaltreue Beobachtung"

Bemerkung:
Es gibt i.a. viele unterschiedliche Linearisierungen!

- wieviele? (Grössenordnung in Abhängigkeit der Ereignis- und Prozesszahl?)
- eine sequentielle Berechnung besitzt offenbar nur eine einzige Linearisierung

---

- kausal unabhängige Ereignisse können stets in unterschiedlicher Reihenfolge wahrgenommen werden
- alle kausaltreuen Beobachtungen sind gleichermassen "wahr"
- alle kausaltreuen Beobachter sind sich bzgl. Kausalitätsrelation einig!

Schnitt aller Linearisierungen einer Halbordnung = Halbordnung
(Theorem von Szpilrajn, 1930)

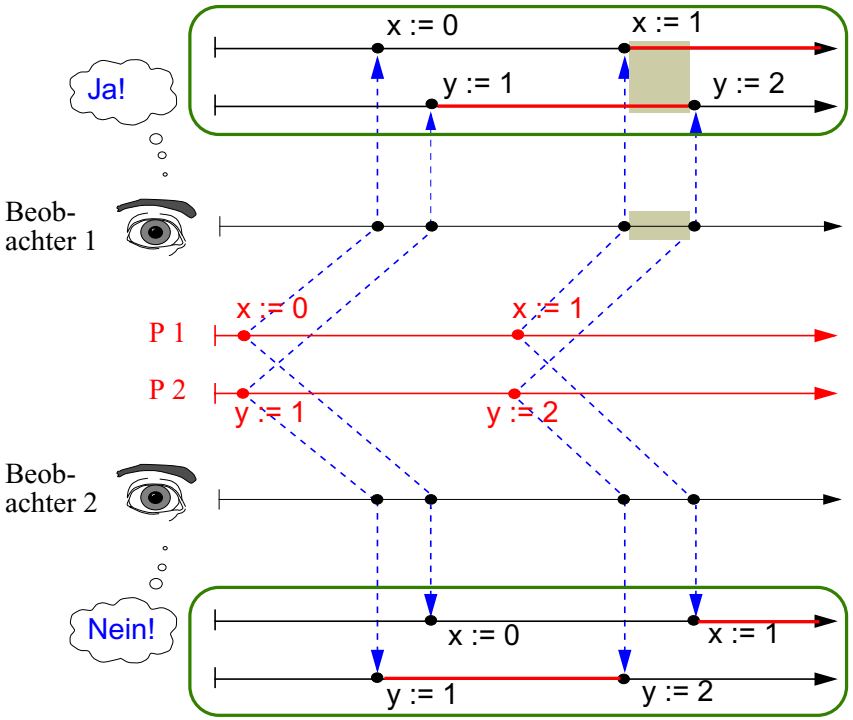
⇒ der "unstrittige Kern" aller Beobachter ist die Kausalrelation der Berechnung selbst!

⇒ in den "wesentlichen" Aspekten stimmen alle Beobachter überein! (die Kausalbeziehung ist also ein beobachterinvariantes, objektives Faktum)

⇒ vert. Berechnung ist durch die Menge aller ihrer Beobachtungen charakterisiert

Das "Entdecken" globaler Prädikate

Frage: Gilt in dieser Berechnung $\Phi \equiv (x = y)$?

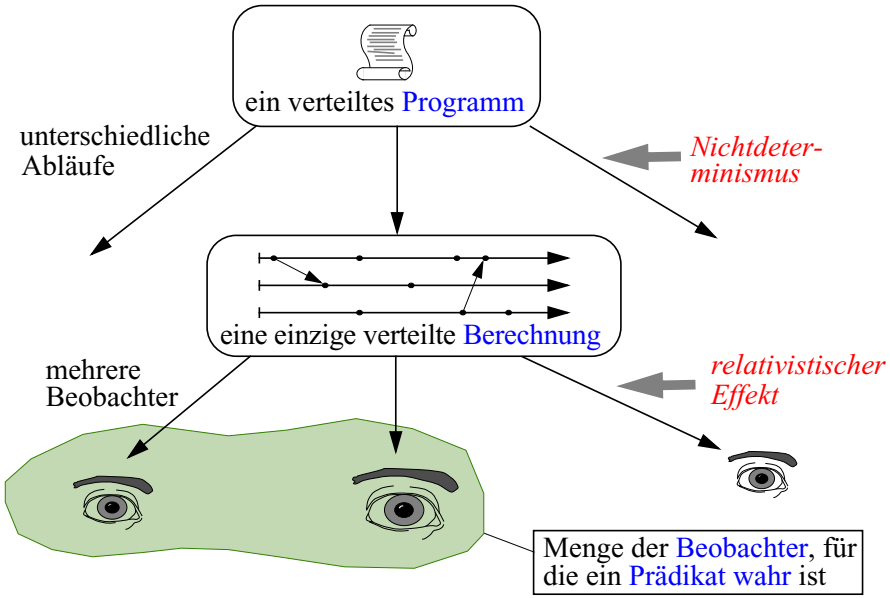


- Meldungen sind alle gleich schnell
- Beide (kausaltreuen!) Beobachtungen sind gleich "richtig"
- Die Beobachter stimmen bzgl. x = y nicht überein!

Aber was denn nun: gilt x=y in dieser Berechnung oder nicht?

"Possible Worlds"

- Verschiedene Beobachter sehen *verschiedene Wirklichkeiten*
  → Frage, ob ein bestimmtes Prädikat gilt, ist u.U. sinnlos!

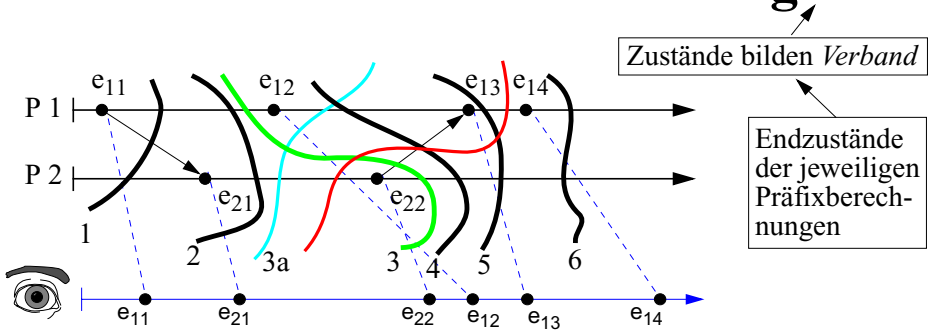


Konsequenz:

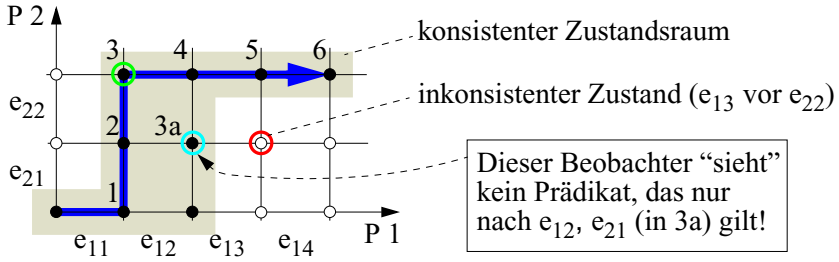
Es ist naiv (d.h. falsch!), einen verteilten Debugger zu entwickeln, mit dem man solche (im sequentiellen Fall "richtigen") Fragen beantworten kann!

- im sequentiellen Fall: Berechnung = Beobachtung
- im verteilten Fall aber: Berechnung ≠ Beobachtung
- Gültigkeit von Prädikaten ist eine Eigenschaft einer Beobachtung, nicht einer Berechnung!
- gibt es sinnvolle beobachterinvariante Prädikate?

Das n-dimensionale Zustandsgitter



- *Beobachtung* =
  - Lineare Folge von Ereignissen
  - Folge damit assoziierter globaler Zustände



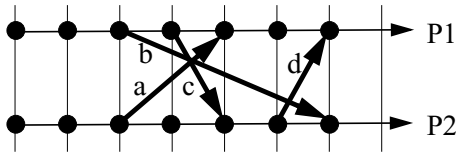
- *Beobachtung* = Pfad von links unten nach rechts oben
  (*kausaltreue Beobachtung,* wenn dieser sich nur im "erlaubten" Bereich aufhält)

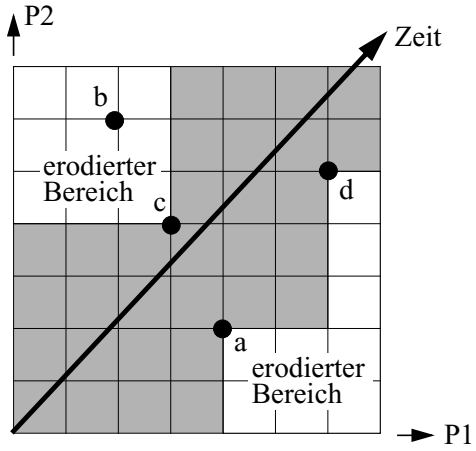
- Ein Beobachter sieht alle *Ereignisse*, aber nur eine Teilmenge aller möglichen *Zustände*!
  → Beobachter kann Prädikate übersehen!

- Mit einem Schnappschussalgorithmus kann nur eine *einzige* (i.a. *lückenhafte*) von vielen Beobachtungen erlangt werden

Der erodierte Zustandshyperwürfel

- Hier: zwei Prozesse → 2-dimensionaler Würfel

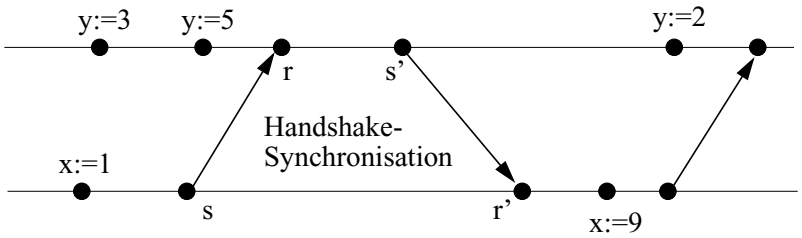




- "Erosion" der inkonsistenten Zuständen von den Ecken her
  - keine Nachricht wird empfangen, bevor sie gesendet wurde
  - ein Prozess blockiert in einer Empfangsanweisung, bis eine Nachricht verfügbar ist (und das zugehörige send somit ausgeführt wurde)
  - Nachrichten zur Synchronisation: sorgen dafür, dass kein Prozess sich zu schnell entfernt (zwingen Prozesse in den "Schlauch" von links unten nach rechts oben)

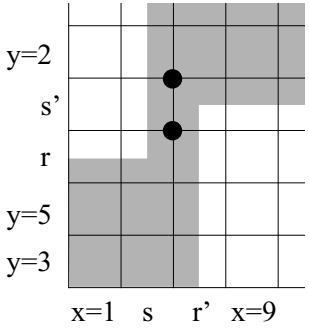
Synchronisationsengpässe

- Hier sieht *jeder* kausaltreue Beobachter "kurzzeitig" y=5, x=1:



- Der Zustand y=5, x=1 wird also auf alle Fälle angenommen; er ist daher "unvermeidlich"

- Verallgemeinerung: "Barrier Synchronisation"
  (erst wenn alle Prozesse die Barriere erreicht haben, dürfen sie weiterlaufen)



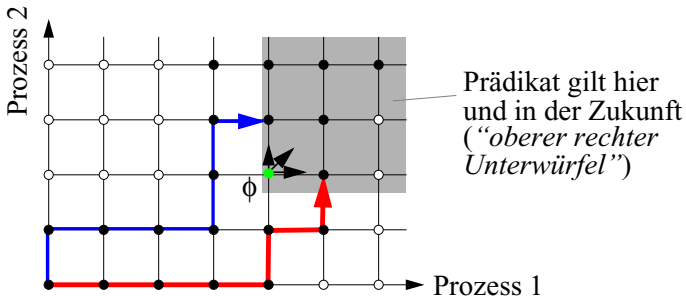
Durch die beiden markierten Zustände muss jeder Beobachter; dort gilt jedoch x=1, y=5

⇒ Mass für die Parallelität bzw. Unabhängigkeit?
(ein gutes numerisches Mass für "Parallelität" zu finden ist schwierig!)

Stabile Prädikate

- Informell: Monoton - "einmal wahr, immer wahr"
- Def.: wenn $c1 < c2$, dann $\Phi(c1) \Rightarrow \Phi(c2)$

Halbordnung im Zustandsverband



- Sind *beobachterunabhängig*
  - jeder Beobachter muss durch den oberen rechten Würfel (Fairness...)
  - lassen sich daher einfach mit einer einzigen Beobachtung feststellen (jede andere Beobachtung wird Φ früher oder später ebenfalls entdecken)

- Für zwei Beobachtungen B_1, B_2 gilt: Falls B_1 Φ "entdeckt", dann gibt es einen *gemeinsamen späteren Zustand* (Verbandseigenschaft!) von B_1, B_2, bei dem Φ gilt
  - spätestens der Endzustand (bei endlichen Berechnungen)
  - B_1 kann z.B. die "echte" Ereignisfolge in Realzeit sein, B_2 eine Beobachtung

- Ein *gelegentlicher* (konsistenter) Schnappschuss genügt!
  - wenn der Schnappschussalgorithmus die Gültigkeit von Φ ermittelt, dann gilt Φ "jetzt" tatsächlich
  - wenn Φ "jetzt" gilt, dann meldet dies ein (jetzt gestarteter) Schnappschussalg.

- Es gibt wichtige stabile Prädikate, z.B. Terminierung, Garbage, Deadlock...
- Aber woher weiss man eigentlich, ob bzw. dass ein Prädikat stabil ist?

Logische Zeit in verteilten Systemen

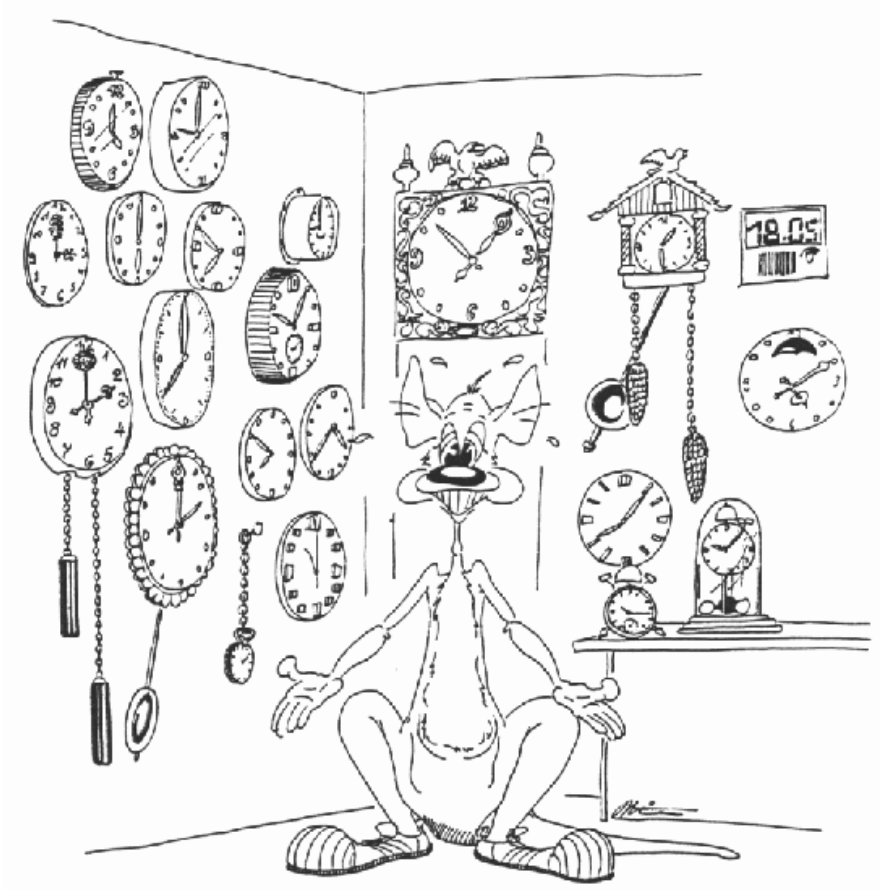
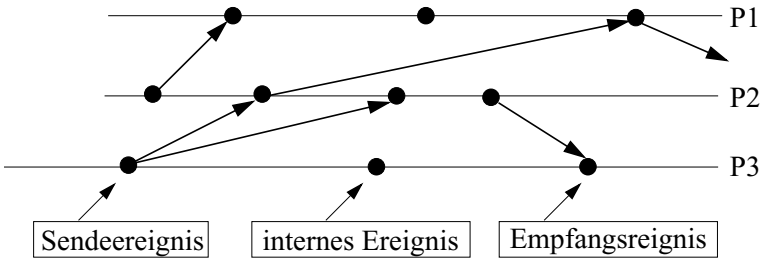


Bild: R. G. Herrtwich, G. Hommel

Kausalrelation ("Happened Before", Lamport 1978)

- *Zur Wiederholung:*



- interessant: von links nach rechts verlaufende "Kausalitätspfade" (bestehend aus Nachrichtenpfeilen + Teilstücken auf Prozessachsen)

- *Kausalrelation* '<' auf der Menge E aller Ereignisse:

"Kleinste" transitive Relation auf E, mit **x** < **y** wenn:

1) **x** und **y** auf dem gleichen Prozess stattfinden und **x** vor **y** kommt, *oder*

2) **x** ist ein Sendeereignis und **y** ist das korrespondierende Empfangsereignis

- In einem Zeitdiagramm gilt für je zwei Ereignisse e, e' die Relation e<e' genau dann, wenn es einen Kausalitätspfad von e nach e' gibt

Logische Zeitstempel von Ereignissen

- Verteilte Berechnung abstrakt: n Prozesse, halbgeordnete Ereignismenge E, Nachrichten (Sende- / Empfangsereignis)
- Zweck: Ereignissen eine Zeit geben ("dazwischen" egal)
- Gesucht: Abbildung C: E → H

  Clock

  "Zeitbereich": Halbgeordnete Menge --> "früher", "später"

- Für e ∈ E heisst C(e) *Zeitstempel* von e
- C(e) bzw. e *früher* als C(e') bzw. e', wenn C(e) < C(e')
- Wie soll H aussehen?

  z.B.:

  - **N** (lineare Ordnung)
  - **R** (bzw. REAL-Datentyp)
  - Potenzmenge von E
  - $\mathbf{N}^n$ (d.h. n-dim. Vektoren)

- Sinnvolle Forderung:

  Kausalrelation ("Pfad im Diagramm")

  Uhrenbedingung: $e < e' \Rightarrow C(e) < C(e')$

  Ordnungshomomorphismus

  Zeitrelation "früher"

Interpretation ("Zeit ist kausaltreu"):

Wenn ein Ereignis e ein anderes Ereignis e' beeinflussen kann, dann muss e einen kleineren Zeitstempel als e' haben

Logische Uhren von Lamport

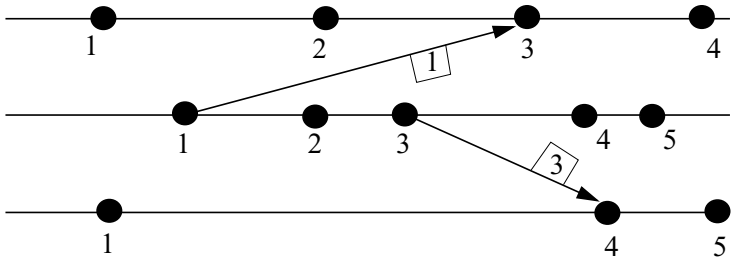
Commun. ACM 1978:
Time, Clocks, and the Ordering of Events in a Distributed System

$C: (E,<) \rightarrow (\mathbf{R},<)$ Zuordnung von Zeitstempeln

Kausal-relation

(oder **N**)

$e < e' \Rightarrow C(e) < C(e')$ Uhrenbedingung



Protokoll zur Implementierung der Uhrenbedingung:

- Lokale Uhr (= Zähler) *tickt* "bei" *jedem* Ereignis
- Sendeereignis: Uhrwert mitsenden (*Zeitstempel*)
- Empfangsereignis: *max*(lokale Uhr, Zeitstempel)

  zuerst! danach "ticken"

Behauptung:
Protokoll respektiert Uhrenbedingung

Beweis: Kausalitätspfade sind monoton...

Kausale Unabhängigkeit

- *Def.* ["kausal unabhängig"]: $a \| b \Leftrightarrow: \neg(a < b) \wedge \neg(a > b)$

  Beachte: $\|$ ist *nicht transitiv*! (Gilt eigentlich $a \| a$?)

- *Behauptung*: $C(a) = C(b) \Rightarrow a \| b$

  Bew.:

  $C(a) = C(b) \Rightarrow \neg(C(a) < C(b)) \wedge \neg(C(a) > C(b))$

  $\Rightarrow \neg(a < b) \wedge \neg(a > b)$

  $\Rightarrow a \| b$ (Def.)

  Verwende $\neg(C(x) < C(y)) \Rightarrow \neg(x < y)$ aus der Uhrenbedingung!

- Gilt die Umkehrung der Uhrenbedingung?

  Nein, $C(e) < C(e') \Rightarrow e < e'$ gilt nicht!

  Es gilt nur: $C(e) < C(e') \Rightarrow e < e'$ *oder* $e \| e'$

  vgl. Beispiel

  Das heisst:

  Aus den Zeitstempeln lässt sich nicht (immer) schliessen, ob zwei Ereignisse kausal voneinander abhängig sind, oder ob sie unabhängig sind!

  $\Rightarrow$ Aber wozu sind solche Zeitstempel dann gut?

Vektorzeit: Motivation

Umkehrung der Uhrenbedingung gilt nicht für Lamport-Zeit

- $C(e) < C(e') \Rightarrow e < e'$ gilt nicht!
- es gilt nur: $C(e) < C(e') \Rightarrow e < e'$ oder $e \| e'$

Zeit := vergangene Zeit

viele Uhren messen die Zeit, indem sie *vergangene* Sekunden zählen

:= Vergangenheit

:= Menge vergangener Ereignisse

vgl. dies mit der *Lamport-Zeit* ("lokal vergangen")

Kausalrelation

Zeit(e) := $\{e' \mid e' \leq e\}$ = *Kegel* von e

Genauer: Zeitstempel eines Ereignisses

Kann durch lokal späteste Ereignisse repräsentiert werden (linksabgeschlossen)

Hiervon gibt es n Stück (n = Anzahl der Prozesse)

!

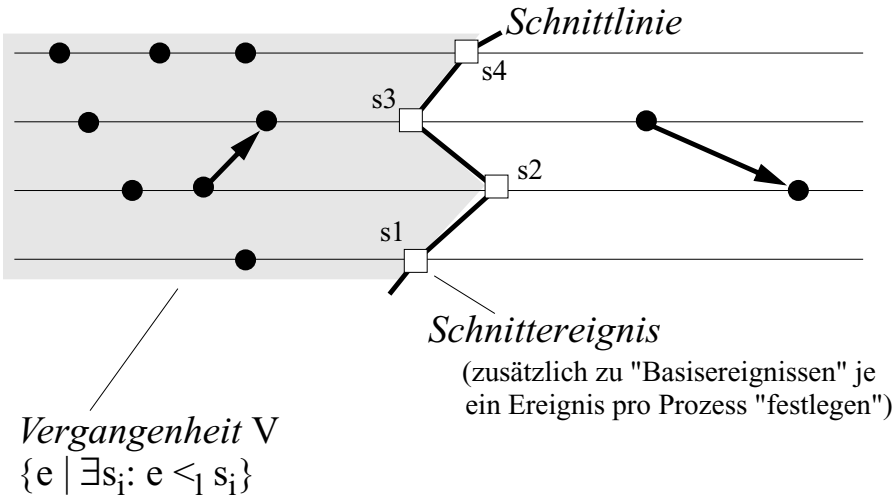
$\rightarrow$ *Zeitstempel* ist n-dimensionaler Vektor

$\rightarrow$ *Zeit* ist Menge n-dimensionaler Vektoren

$\rightarrow$ *Uhr* ist ein array C[1:n]

zum Anzeigen von Zeitvektoren

Schnitt und Schnittlinie



- Schnittlinie trennt Zeitdiagramm / Ereignismenge in zwei disjunkte Mengen "Vergangenheit" / "Zukunft"

- Bemerkung: $e \in V \wedge e' <_l e \Rightarrow e' \in V$
  (linksabgeschlossen bzgl. lokaler Kausalrelation)

Denkübung: Man vergleiche den Begriff des *Schnittes* (insbes. des *konsistenten Schnittes*, vgl. nächste Seite) mit dem früher erwähnten Begriff der *Präfixberechnung*! Man beachte auch die Halbordnungsstruktur bzw. Verbandsstruktur dieser Begriffe.

Konsistente Schnitte

Def. **Schnitt**:

$S \subseteq E$ heisst *Schnitt* von E, falls $e \in S \wedge e' <_l e \Rightarrow e' \in S$
(d.h. Schnitt wird mit seiner Vergangenheit identifiziert)

Def. **konsistenter Schnitt**:

$S \subseteq E$ heisst *konsistent*, falls $e \in S \wedge e' < e \Rightarrow e' \in S$

Def.: Schnitt S *später* als S' $:\Leftrightarrow$ $S' \subseteq S$

bzw. $\subset$ bei "strikt später"

Schreibweise auch: S' < S

Beh.: Jeder konsistente Schnitt ist ein Schnitt

Bew.: $<_l \subseteq <$

Bem.: Schnitt(linie) inkonsistent $\Leftrightarrow$
$\exists$ "Nachricht aus der Zukunft"

Bem.: Schnitt(linie) konsistent $\Leftrightarrow$
Schnittereignisse paarweise kausal unabhängig

Bew. als Übung

Bem.: Schnitt(linie) konsistent $\Leftrightarrow$
lässt sich senkrecht darstellen (Gummibandtransf.)

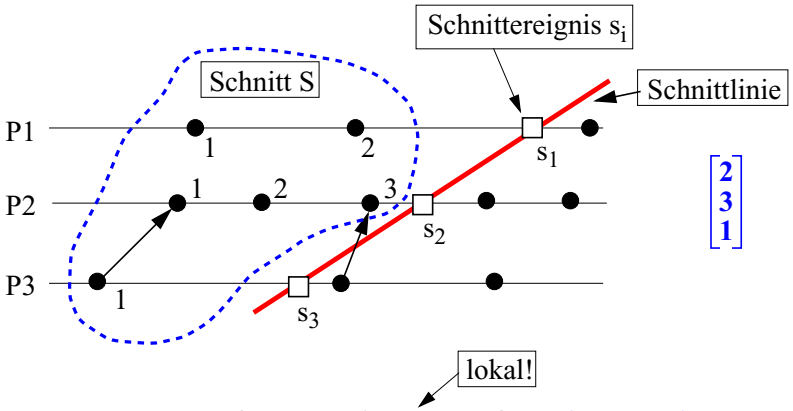
Bew. bereits bekannt: Diagramm auseinanderschneiden und versetzen

Vektorzeitstempel von Schnitten

Zeitstempel $\tau(S)$ eines Schnittes S ist ein Vektor aus $\mathbf{N}^n$

alle Ereignisse, die links von einer Schnittlinie liegen

Anzahl der Prozesse



Def. $\tau(S)[i] := |\{e \in E_i \mid e <_1 s_i\}| := |S \cap E_i|$

(für jede Komponente i mit $1 \le i \le n$)

Interpretation:

$\tau(S)[i]$ ist die "Stelle", wo die Prozessachse von P_i durch die Schnittlinie geschnitten wird

Beachte: man kann zu *konsistenten* und *inkonsistenten* Schnitten den zugehörigen Zeitvektor definieren!

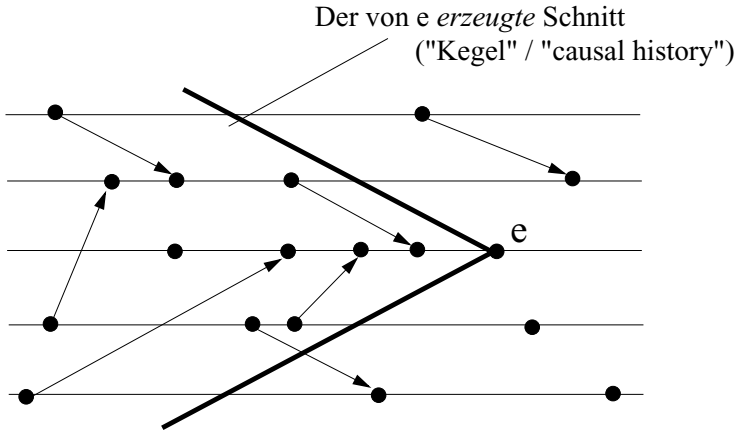
Kausale Vergangenheit eines Ereignisses

Def. *kausale Vergangenheit* $\downarrow(e)$ eines Ereignisses e:

$$\downarrow(e) = \{e' \mid e' \le e\}$$

Beh.: $\downarrow(e)$ ist ein konsistenter Schnitt

Bew. als Übung



Beh.: $e' \le e \Leftrightarrow e' \in \downarrow(e)$

Beh.: $e \parallel e' \Leftrightarrow \neg(e \in \downarrow(e')) \wedge \neg(e' \in \downarrow(e))$

(Bew. klar)

Vektorzeitstempel von Ereignissen

Kausale Vergangenheit

- Bezeichne $\downarrow e$ den *Kegel* $\{e' \in E \mid e' \leq e\}$ von e

  - jeder Kegel ist ein konsistenter Schnitt (da linksabgeschlossen) ("Kegelmantel" könnte also als senkrechte Linie gezeichnet werden!)

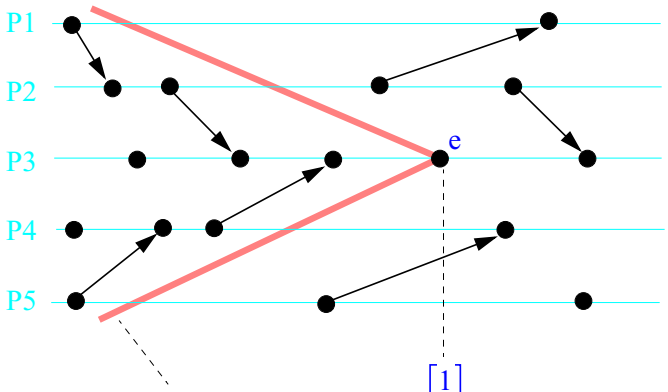
  - Repräsentation durch die n lokal am weitesten rechts liegenden Ereignisse

- Dann definiere $\tau(e) := \tau(\downarrow e)$

Menge der Ereignisse von P_i

Also: $\tau(e)[i] := |\{e' \in E \mid e' \leq e\} \cap E_i| := |\{e' \in E_i \mid e' \leq e\}|$

  - das heisst: Zeitstempel eines Ereignisses = Zeitstempel seines Kegels

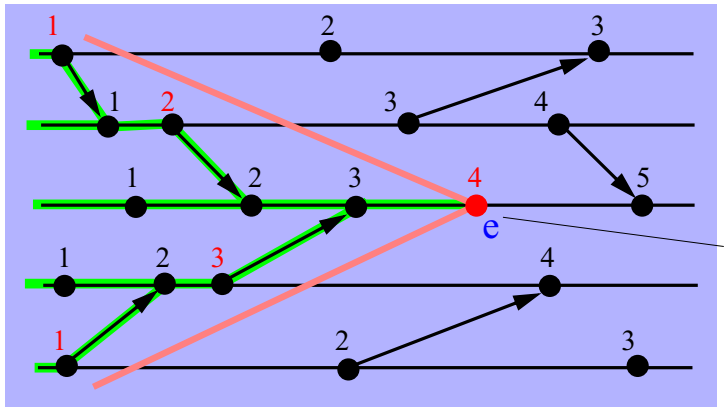


Schnittlinie in der Form eines Kegels (vgl. "Lichtkegel" der relativistischen Physik): $\downarrow e$ ist ein konsistenter Schnitt

$\begin{bmatrix}1\\2\\4\\3\\1\end{bmatrix}$ $\tau(e)$: Vektorzeitstempel von e

$\tau(e) < \tau(e')$

- Jeder Prozess numeriert seine Ereignisse lokal durch



- Vektor $\tau(e)$ repräsentiert gesamte kausale Vergangenheit des Ereignisses e

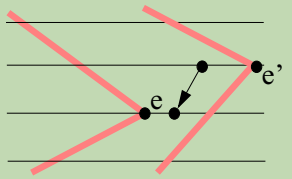
  - Veranschaulichung durch einen "Kegel" $\{e' \mid e' \leq e\}$

  - Nummer des jeweils lokal letzten kausal vorangehenden Ereignisses steht in der jeweiligen Komponente des Vektors

- Interpretation von $\tau(e) < \tau(e')$:

  - e liegt in der kausalen Vergangenheit von e'

  - Kegel von e ist ganz im Kegel von e' enthalten



Def. "Zeitstempelarithmetik”

$$\begin{bmatrix}1\\3\\4\\3\\2\end{bmatrix} \leq \begin{bmatrix}1\\7\\4\\6\\2\end{bmatrix}$$

vergleichbar
(komponentenweise ≤)

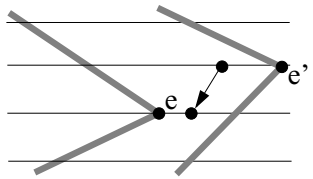
$$\begin{bmatrix}1\\3\\4\\3\\7\end{bmatrix} \parallel \begin{bmatrix}5\\3\\8\\3\\2\end{bmatrix}$$

"konkurrent"

‘<‘ definiert als “≤ aber ≠”

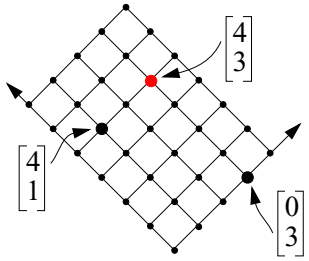
Interpretation von $\tau(e) < \tau(e')$:

- e liegt in der kausalen Vergangenheit von e’
- Kegel von e ist im Kegel von e’ enthalten



$$\sup\left(\begin{bmatrix}1\\4\\2\\3\\7\end{bmatrix}, \begin{bmatrix}8\\3\\4\\3\\2\end{bmatrix}\right) = \begin{bmatrix}8\\4\\4\\3\\7\end{bmatrix}$$

sup = komponentweises Maximum



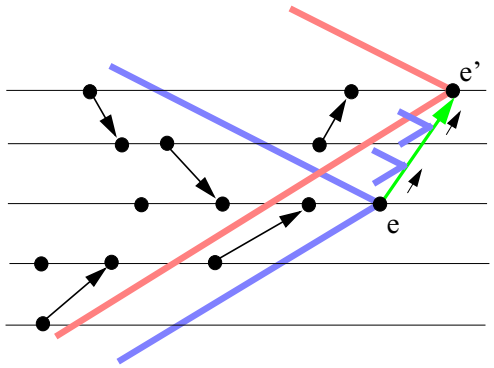
Implementierung der Vektorzeit

- Idee: Analog zur Lamport-Zeit
  (hier allerdings stets vektoriell!)

- Nachrichten enthalten die gesamte kausale Vergangenheit des Senders ⇒ Zeitvektor des Sendeereignisses

- Bei Empfang einer Nachricht:
  - Vereinigung der Kegel
  - ⇒ Supremum der Vektoren

Wissen über vergangene Ereignisse vereinigen



Mitschleppen des Kegels des Sendeereignisses und Vereinigung mit dem Kegel des Empfangsereignisses

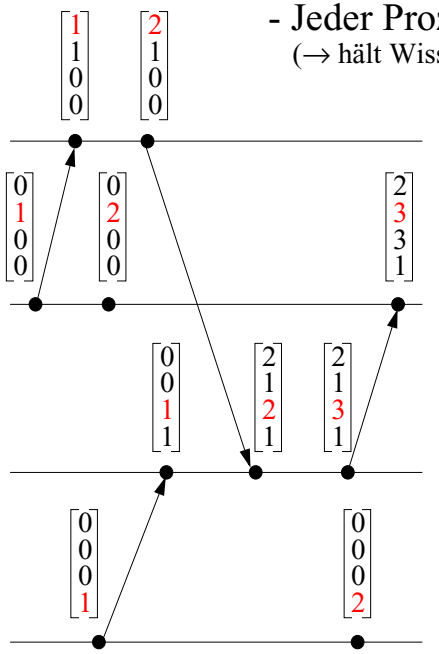
→ "induktiv": ein Ereignis hat ein "vollständiges Wissen" über alle seine vergangenen Ereignisse

Propagieren des Zeitwissens

Andere Zeiten, andere Sitten

(→ Implementation der Vektorzeit)

- Jeder Prozess besitzt eine *Vektoruhr*
  (→ hält Wissen über vergangene Ereignisse)



- *bei jedem Ereignis:*
  eigene Komponente erhöhen
- *beim Senden:*
  neuen Vektor mitsenden
- *beim Empfangen:*
  komponentenweises Maximum der beiden Vektoren

Vereinigung der beiden Kegel

Kausalrelation

bzgl. Zeitvektor

- *Behauptung*: $e < e' \Leftrightarrow \tau(e) < \tau(e')$

- *Anschauliche Interpretation*:

monoton bzgl. Zeitvektoren!

  - $\tau(e) \leq \tau(e') \Leftrightarrow$ es gibt eine Kausalkette von e zu e'

- *Korollar*: $e \parallel e' \Leftrightarrow \tau(e) \parallel \tau(e')$

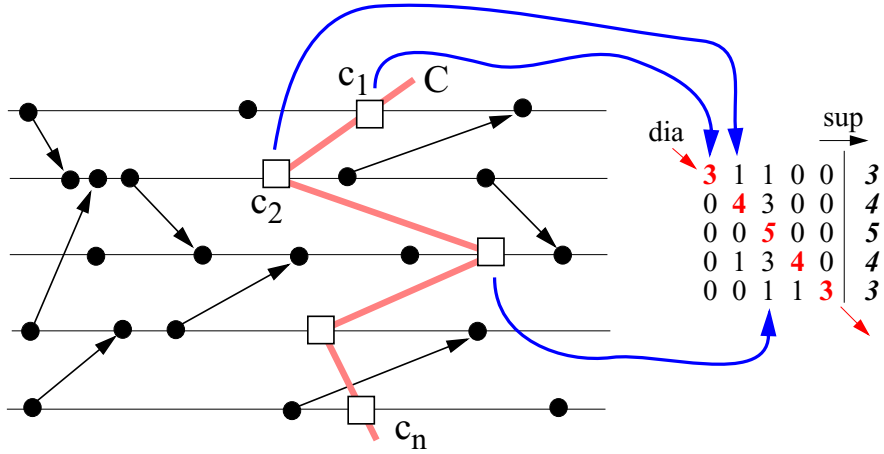
Interpretation: Genau die "gleichzeitigen" Ereignisse beeinflussen sich nicht ggs.

Schnittmatrix

- Schnittmatrix $ eines Schnittes C (mit Schnittereign. c_i):

$$\$(C) := (\tau(c_1), \tau(c_2),...,\tau(c_n))$$

d.h. Schnittereignisvektoren c_i als Spaltenvektoren



Frage: Kann man an den Schnittmatrizen etwas über die Schnitte erkennen? (z.B.: ob später, früher; ob konsistent...)

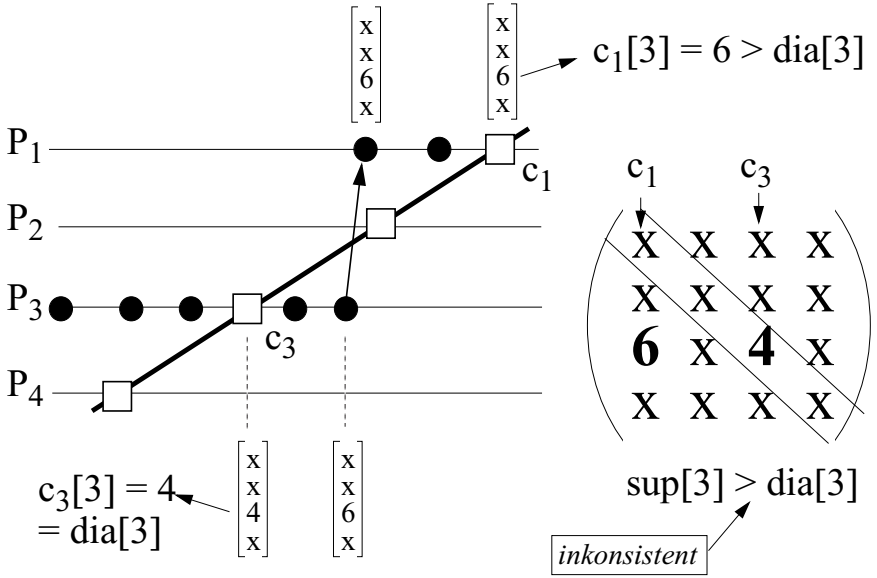
$$C \text{ konsistent} \Leftrightarrow \text{dia}(\$) = \sup(\$)$$

Diagonalvektor

Zeilenmaximum

(d.h. Maximum einer Zeile ist das Diagonalelement)

Das “sup = dia”-Konsistenzkriterium



Ein Prozess (P_1) verschieden von P_3 weiss (bei c_1) etwas über lokale Ereignisse auf P_3, von denen P_3 selbst noch nichts weiss (d.h. die *nach* c_3 geschehen)

⇔

Es gibt einen Pfad von einem Ereignis auf P_3 *nach* c_3 zu einem Ereignis *vor* c_1

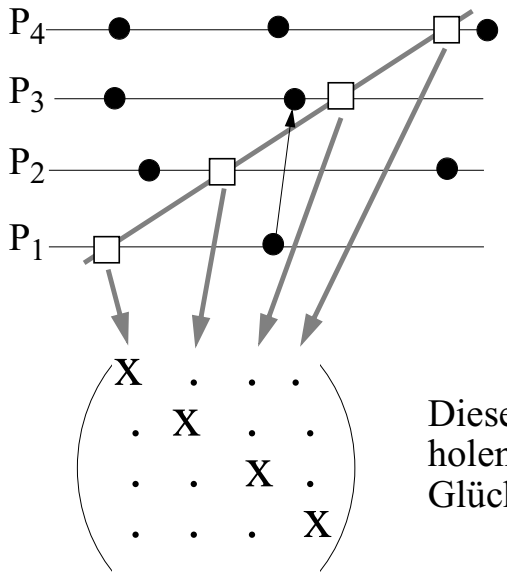
⇔

[Generalisierung über alle Indizes i≠j]

Der Schnitt ist inkonsistent

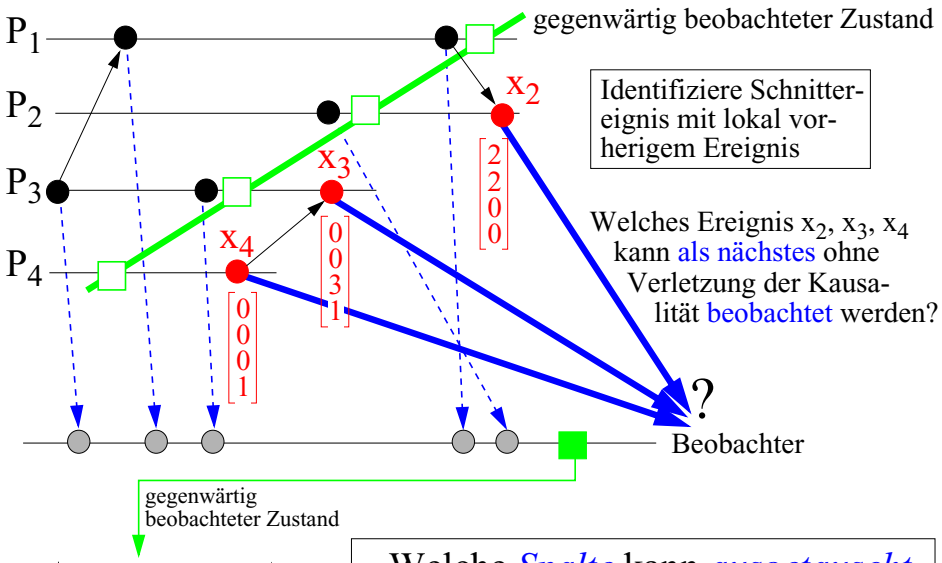
Implementierung konsistenter Schnappschüsse mit Vektorzeit?

Ein erster Ansatz: Alle Prozesse auffordern, ihren lokalen Zustand zu senden und testen, ob konsistent:



Dieses solange wiederholen, bis man einmal Glück hat...

Realisierung kausaltreuer Beobachter



$$\begin{pmatrix} \mathbf{2} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 1 & 0 & \mathbf{2} & 0 \\ 0 & 0 & 0 & \mathbf{0} \end{pmatrix}$$

- Welche *Spalte* kann *ausgetauscht* werden? (x_2, x_4, aber nicht x_3)
- Beobachter merkt sich dia($); es muss *Zeitstempel* ≤ *dia($)* sein (ausgenommen die Diagonalkomponente)

- Strategie: dia($) = sup($) → stets konsistent halten!
- Beobachter benötigt nur den Diagonalvektor, keine Matrix, um momentanen Zustand zu identifizieren

---

Prinzip: Verwende Vektorzeit um (indirektes) *Wissen* über "kausal frühere" Sendeereignisse zu kodieren:

"Dieses Ereignis hängt von einem anderen ab, das ich eigentlich erhalten haben müsste; also warte ich das andere erst ab..."

Resümee: Konzepte der Vorlesung

- Prinzipielle Phänomene und Begriffe herausarbeiten
  - Kausalität, Konsistenz, verteilte Berechnung, safety und liveness,...
- Geeignete Modelle und Abstraktionen entwickeln
  - z.B. Zeitdiagramme, Atommodell, Zustandsgitter, Gummibandtransform.
- Problemlösungs-, Analyse- und Verifikationstechniken
  - z.B. Beweis über Invarianten
- Techniken, Einsichten, Zusammenhänge,...
  - Komplexitätsanalyse
  - Transformationen zwischen Problemklassen
  - Problemverständnis von einem höheren Standpunkt

Resümee: Themen der Vorlesung

- Beispiele für verteilte Berechnungen und Algorithmen
  - verteilte ggT-Berechnung
  - verteiltes Lösen von Zahlenrätseln
  - verteilte Approximation
- Grundalgorithmen
  - Flooding
  - Echo-Algorithmus (Wellenalgorithmus, Spannbaum)
- Verteilte Terminierung
  - Doppelzählverfahren
  - Zeitzonenverfahren
  - für synchrone Kommunikation: DFG-Verfahren
  - Kreditmethode

Grundphänomen "inkonsistenter Sicht"; nur problemspezifische Lösungen dafür

- Wechselseitiger Ausschluss
  - Grundprinzipien
  - Maekawa
  - Token-basierte Verfahren

Synchronisation in vert. Sys. (viele wollen, einer darf; Sicherheit, Deadlockfreiheit, Fairness)

- Election
  - Chang/Roberts-Verfahren (Ring); bidirektionale Varianten
  - Hirschberg/Sinclair und Peterson's Algorithmen: O(n log n) worst case
  - Election auf Bäumen
  - untere Schranke O(e) für Nachrichtenkomplx. bei allg. Netzen
  - Election in anonymen Netzen (probabilistische Algorithmen)

Symmetriebrechung in vert. Sys.: verteilte Wahl eines "Repräsentanten"

Resümee: Themen (2)

- Garbage-Collection
  - Mutator, collector, Formalisierung
  - Behind the back copy
  - Verteiltes Garbage-Collection
  - Referenzzähler (verschiedene Lösungen; z.B. WRC, LRC)
  - Implementierungstechniken
- Garbage-Collection $\Rightarrow$ Terminierungserkennung
- Wellenalgorithmen
  - Eigenschaften, Spannbäume,...
- Sequentielle Traversierungsverfahren
  - Methode von Tarry (1895)
- Parallele Traversierungsverfahren
  - Verteilen von Information ("flooding"), Echo-Algorithmus

Resümee: Themen (3)

- Schnappschuss, Konsistenz, Beobachtungen, Prädikate,...
  - Kausalrelation, kausale Vergangenheit...
  - Halbordnung, Verband,...
  - Schnitt, globaler Zustand
  - Kausal konsistente Beobachtung
  - Globale Prädikate, stabile Prädikate
  - Schnappschussalgorithmen

- Logische Zeit
  - Uhrenbedingung
  - Lamport-Uhren
  - Vektorzeit: Eigenschaften und Implementierung
  - Schnittmatrix
  - Konsistenzkriterium
  - Implementierung von kausaltreuen Beobachtern