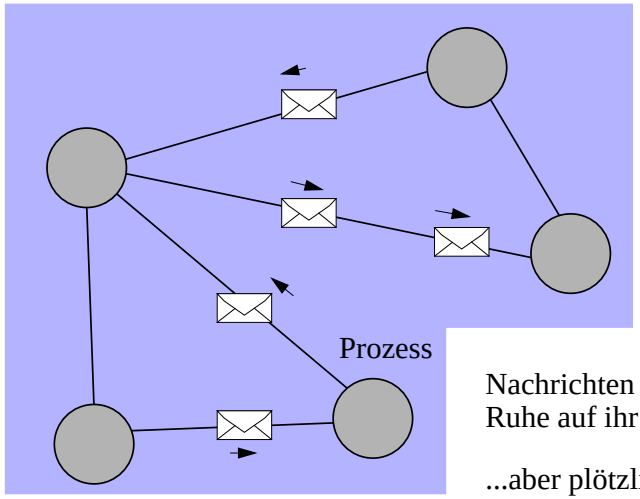
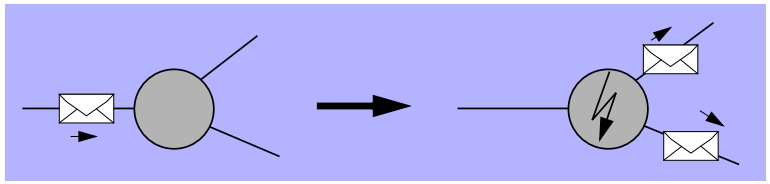
Globale Sicht ’atomarer’ Berechnungen



idealisierter Beobachter

Nachrichten fliessen in aller Ruhe auf ihr Ziel zu...

...aber plötzlich "explodiert" ein Prozess, wenn er von einer Nachricht getroffen wird!

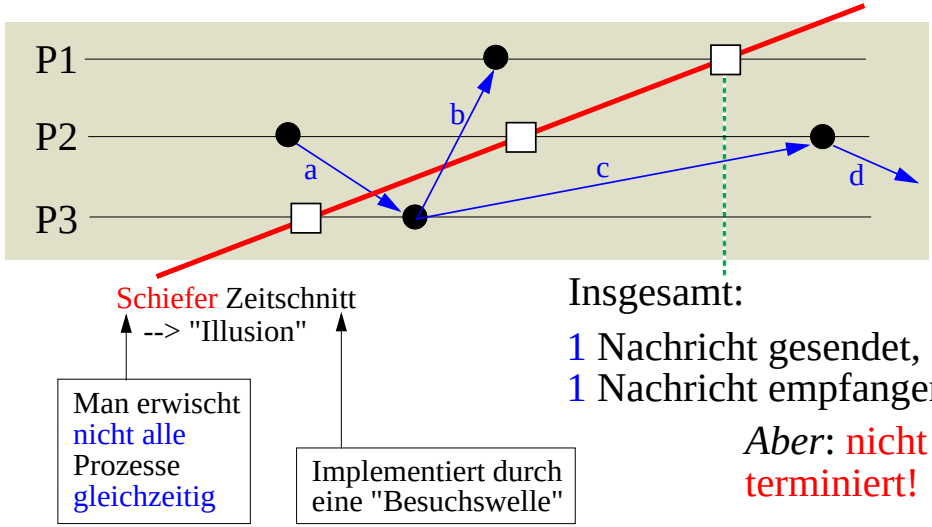


Terminiert, wenn in der globalen Sicht kein ✉ existiert

- Statt im "passiv/aktiv-Modell" genügt es offenbar, im Atommodell die Terminierungserkennung zu lösen (wieso?)
- Wie sehen die Aktionen der Basisberechnung in diesem Modell aus?

Verteilte Terminierung: Lösungen durch Zählen von Nachrichten?

- Genügt das (verteilte) Zählen von gesendeten und empfangenen Nachrichten?
- Einfaches Zählen genügt nicht, Gegenbeispiel:



Insgesamt:

1 Nachricht gesendet,
1 Nachricht empfangen.

Aber: nicht terminiert!

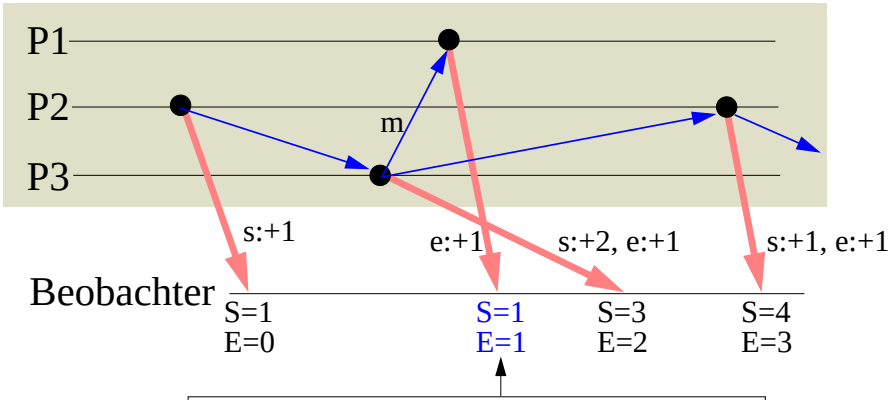
Ursache (informell):

- Nachricht aus der "Zukunft"
  - kompensiert die Zähler
- Inkonsistenter Schnitt
  - ist nicht äquivalent zu einem senkrechten Schnitt

Lösung durch "Ursachenvermeidung"? Ideen vielleicht:

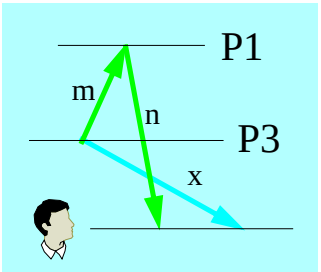
- Nachrichten aus der Zukunft *vermeiden* oder zumindest *erkennen*?
- Senkrechten Schnitt simulieren durch *Einfrieren* der Prozesse?

Beobachter über gesendete und empfangene Nachrichten informieren?



- Gleiches Szenario wie eben: Beobachter erfährt, dass m *empfangen* wurde, aber nicht, dass m *gesendet* wurde!

- Man beachte auch, dass hier eine Nachricht (x) in indirekter Weise (via m und n) "überholt" wurde!



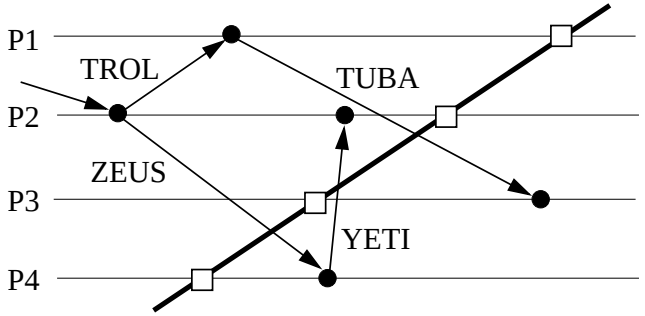
Vermutung: Wenn Informationsnachrichten nicht (indirekt) überholt werden können, dann kann das Phänomen eines "schiefen Bildes" nicht auftreten!

- worauf gründet sich die Vermutung?
- kann man solchermassen korrekte ("kausaltreue") Beobachtungen erzwingen?

Nachrichten eindeutig benennen?

Prinzip: Jede Nachricht bekommt einen (global) eindeutigen Namen:

- TROL, ZEUS, TUBA, YETI,... (?)
- Nachricht kennt ihren Namen
- Sender weiss, welche Nachrichten gesendet wurden
- Empfänger weiss, welche Nachrichten empfangen wurden



ZEUS und TUBA müssen Schnitt von links nach rechts überquert haben, YETI von rechts nach links (relativ zum Schnitt!)

- Welle akkumuliert Namen der gesendeten und Namen der empfangenen Nachrichten
- Wenn eine gesendete nicht empfangen wurde, muss sie den Schnitt überquert haben ==> Terminierung nicht melden
- Terminiert, wenn alle "bekanntermassen gesendeten" auch empfangen wurden? (Beweis?)
  - Tip: Wenn keine Nachricht den Schnitt (von links nach rechts??) überquert, ist der Lebensfaden des Systems gerissen; rechts des Schnittes kann dann keine Aktivität mehr entfacht werden (wieso?)

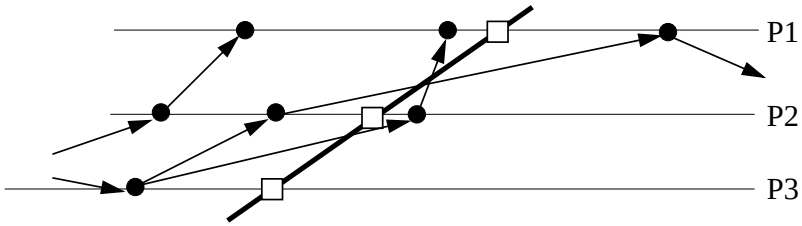
Frage: Wie geht das ganze überhaupt initial los?

Eindeutige Nachrichtennamen?

- Sender könnte Nachrichten fortlaufend numerieren und seinen eigenen eindeutigen Namen hinzufügen
  - lässt sich einfacher verwalten als beliebige (global eindeutige) Namen
- Es genügt wohl auch eine fortlaufende Numerierung pro Sender-Empfänger-Beziehung ("Kanal")
  - z.B. 17.4.239 ("239. Nachricht von Knoten 17 an Knoten 4")
  - Verwaltungsaufwand ist recht hoch (bei FIFO benötigt man keine Mengen, es genügen $O(n^2)$ Zähler)

Genügt pauschales Zählen pro Kanal?

(anstatt Nachrichten pro Kanal individuell zu betrachten)

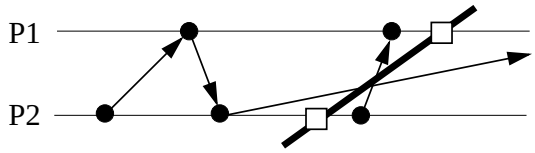


- Welle stellt folgendes fest:
  - auf Kanal P2P1 sind 2 Nachrichten gesendet und 2 Nachrichten empfangen worden
  - dennoch überquerte eine Nachricht den Schnitt von P2 nach P1!
- Denkübung: Wäre das bei FIFO-Kanälen korrekt?
  - d.h. wäre dann bei ausgeglichenen Kanalzählern keine Nachricht auf diesem Kanal unterwegs?

Behauptung (auch bei non-FIFO!):

Wenn entlang eines Schnittes *alle* Kanalzähler bzgl. send/receive ausgeglichen sind, dann überquert keine Nachricht den Schnitt

- Wieso? (intuitives Argument?)
- Beweis?



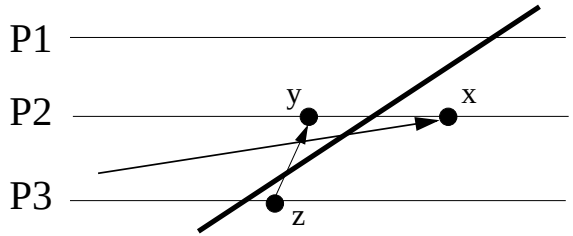
wieso ist das eigentlich kein Gegenbeispiel?

Beweisskizze für das Kanalzählerkriterium

Behauptung: Wenn entlang eines Schnittes pro Kanal gleich viele Nachrichten gesendet wie empfangen wurden, dann ist die Berechnung terminiert

Betrachte frühestes Ereignis (x) *nach* dem Schnitt:

Bei globaler (von links nach rechts fliessender) *Zeit* in der Abb. ist dies klar; wenn man ohne solche graphischen Veranschaulichungen auskommen will, muss man statt dessen die *Kausalrelation* bemühen!

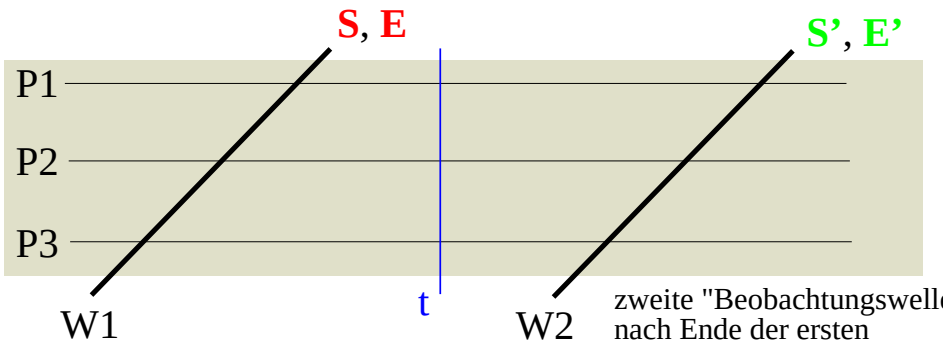


Wir zeigen durch *Widerspruch*: es gibt kein frühestes Ereignis nach dem Schnitt ==> Terminierung

- Dies ist ein Ereignis mit Empfang einer Nachricht, deren Sendeereignis *links* des Schnittes liegt
- Zugehöriger Kanalzähler kann nicht getäuscht werden, da für eine *Kompensationsnachricht* gilt: Empfangen (y) vor dem Schnitt, gesendet (z) danach
- Sendeereignis der Kompensationsnachricht wäre *früheres* Ereignis *nach* dem Schnitt ==> *Widerspruch*
  - Senden ist immer früher als das Empfangen einer Nachricht!
  - z früher als y, y früher als x ==> z früher als x

Zählen pro Kanal ist aber etwas aufwendig ($O(n^2)$ Zähler); geht es nicht doch mit "ganz pauschalen" Zählern?

Das Doppelzählverfahren



d.h. keine Nachricht unterwegs

Behauptung: S=E=S'=E' ==> terminiert

Beweis (Skizze; lässt sich auch formalisieren):

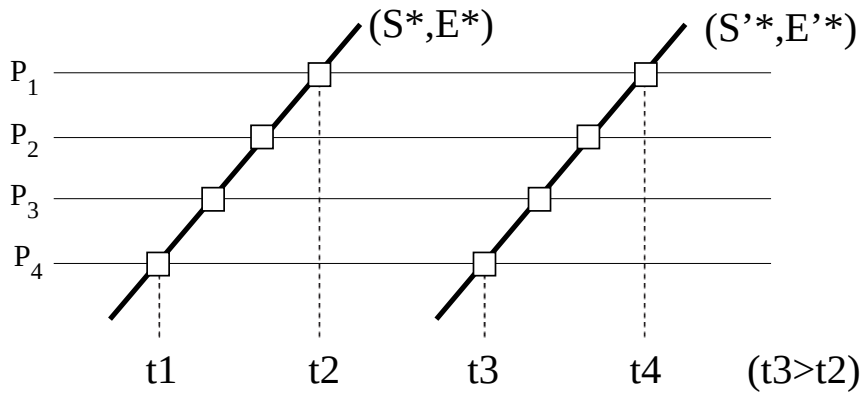
S=S' ==> Keine Nachricht zwischen W1, W2 gesendet.
E=E' ==> _____ "" _____ empfangen.
==> Werte bei t = Werte von W1.

Also: S=E ==> zum globalen Zeitpunkt t gilt:

Anzahl gesendeter = Anzahl empfangener Nachrichten
==> zum Zeitpunkt t ist keine Nachricht unterwegs
==> zum Zeitpunkt t terminiert
==> Berechnung war nach W1 terminiert □

Es gelingt also, für einen bestimmten Zeitpunkt eine kausaltreue Beobachtung (als senkrechten Schnitt) im Nachhinein zu *rekonstruieren*!

Formalerer Beweis des Verfahrens



Notation:

- Lokaler Send-Zähler des Prozesses P_i zur Zeit t: $s_i(t)$
- Lokaler Empf.-Zähler des Prozesses P_i zur Zeit t: $e_i(t)$
- $S(t) := \sum s_i(t)$ $E(t) := \sum e_i(t)$

Lemmata:

(1) $t \le t' \Rightarrow s_i(t) \le s_i(t'), e_i(t) \le e_i(t')$ [Def.]
(2) $t \le t' \Rightarrow S(t) \le S(t'), E(t) \le E(t')$ [Def., (1)]
(3) $E^* \le E(t2)$ [(1), e_i wird "eingesammelt" vor t2]
(4) $S'^* \ge S(t3)$ [(1), s_i wird "eingesammelt" vor t3]
(5) Für alle t: $E(t) \le S(t)$ [Induktion über die atomaren Aktionen]

Beweis:

$E^* = S'^* \Rightarrow E(t2) \ge S(t3)$ [(3), (4)]
$\Rightarrow E(t2) \ge S(t2)$ [(2)]
$\Rightarrow E(t2) = S(t2)$ [(5)]
$\Rightarrow$ terminiert zum Zeitpunkt t2 □

Zwei Zähler genügen!

Anzahl der "in-transit" Nachrichten bei t2 = 0

Eigenschaften des Doppelzählverfahrens

- Vergleich des Empfangszählers der ersten Welle mit mit dem Sendezähler der zweiten Welle (d.h. E = S') ist ein hinreichendes Kriterium

- Falls Terminierung nicht entdeckt wird: Benutze zweite Welle der vorherigen Runde als erste Welle der neuen Runde

---

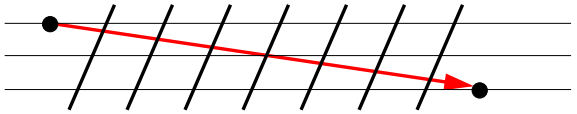
"ablaufinvariant"

- Algorithmus ist *reentrant*: Lokaler Zustand des Prozesses wird durch Kontrollalgorithmus nicht geändert
  ==> jeder Prozess kann unabhängig eine eigene / neue Kontrollrunde starten ("symmetrischer Algorithmus")

- Viele Varianten (zugrundeliegender Wellenalgorithmus)

- Anzahl der Kontrollrunden ist a priori nicht durch die Anzahl der Basisnachrichten begrenzt

  - es kann eine ganz langsame Basisnachricht geben, während der beliebig viele Kontrollrunden gestartet werden können



  - allerdings ist folgende Variante denkbar: ein Prozess, der eine Basisnachricht erhält ohne eine neue auszusenden, startet eine Doppelrunde (Zeitkomplexität?)

Safety- und Liveness-Eigenschaften

Safety: Something bad will never happen...
(Typisch: "für alle eingenommenen Zustände gilt...")

- Bsp.: Nie mehr als 1 Prozess im kritischen Abschnitt
- Bsp.: Variable x wird nie negativ
- Bsp.: Invariante wird nicht verletzt

oft auch "progress"

schliesslich

Liveness: Something good will eventually happen...
(Typisch: "es wird ein Zustand eingenommen, so dass...")

- Bsp.: Variable x wird schliesslich positiv
- Bsp.: Programm terminiert
- Bsp.: erfolgte Terminierung wird schliesslich auch gemeldet

Korrektheit: Algorithmus erfüllt Safety *und* Liveness

Beispiel verteilte Terminierung:

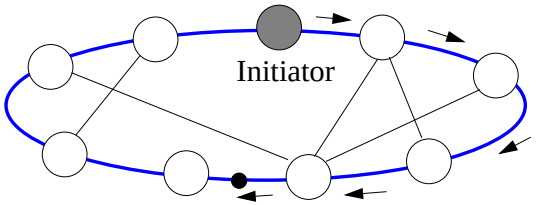
- aufgesetzter Kontrollalgorithmus
  - sagt "ja", wenn Basisberechnung terminiert ist
  - sagt "weiss nicht" sonst ("nein" geht nicht!)
- safe, aber nicht live: meldet stets "weiss nicht"
- live, aber nicht safe: meldet stets "ja"

==> "Kunst": Algorithmus, der safe *und* live ist!
==> Es ist stets safety *und* liveness zu zeigen!

Kontrolltopologien

- Die für den Terminierungsalgorithmus benötigten "Wellen" können unterschiedlich realisiert werden:

1.) *Ring / Hamilton'scher Zyklus:*



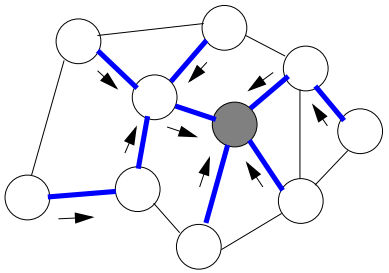
Kontrollnachricht ("Token") besucht zyklisch zwei Mal alle Prozesse und akkumuliert dabei die Zählerstände

"Logischer" Ring genügt!

- in einem zusammenhängenden ungerichteten Graphen kann ein logischer Ring immer gefunden werden, indem man einen Spannbaum "umfährt"
- Zeit- und Nachrichtenkomplexität: O(n)

2.) *Spannbaum:*

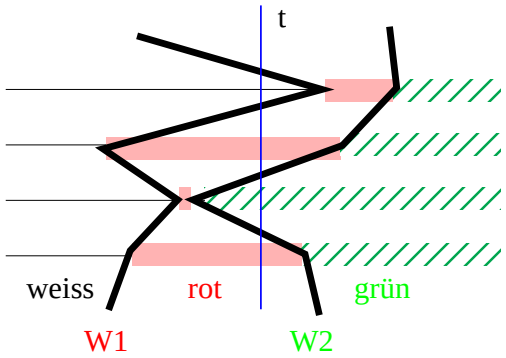
- vereinfachter Echo-Algorithmus: an den Blättern reflektierte Welle sammelt die Zählerstände in akkumulierender Weise ein



- auch hier werden *zwei* solche zum Initiator hinfliessende Wellen benötigt
- bei nicht-entartetem Spannbaum: Viele Nachrichten parallel unterwegs ==> bessere Zeitkomplexität!

Echo-Algorithmus für die beiden Wellen des Doppelzählverfahrens?

- Anwendbar für beliebige zusammenhängende Topologien
- Ausnutzen der beiden "Halbwellen" eines einzigen Laufs des Echo-Algorithmus für die beiden Kontrollrunden!

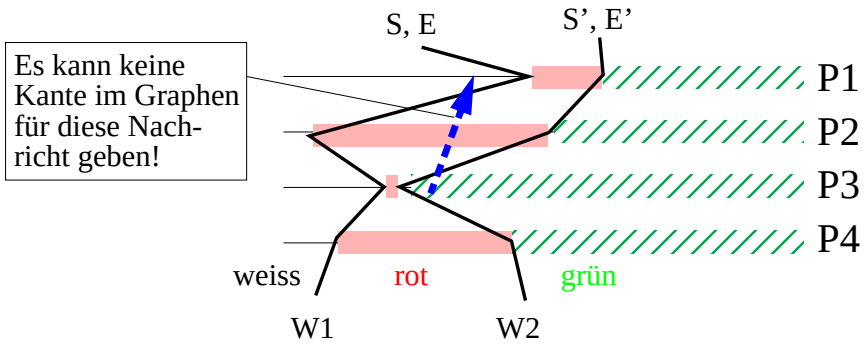


Der Echo-Algorithmus wird als *Transportdienst* zur Realisierung von *zwei* Wellen benutzt, wobei durch die Echo-Nachrichten sowohl die bei W1 lokal gemerkte Information als auch die bzgl. W2 relevante lokale Information an den Initiator übermittelt wird.

- Welle W1: "rot werden"; Welle W2: "grün werden"

- *Behauptung*: Das so realisierte Doppelzählverfahren ist korrekt
- *Problem*: Formeller oder informaler Beweis lassen sich so nicht anwenden, da sich W1 und W2 *überlappen*!

Bemerkung:
Auf vorhandenen Spannbäumen kann man aber *nicht* einfach die vom Initiator ausgesendete Hinwelle und die reflektierte Rückwelle verwenden!



- "Trick": Es gibt keine nach W2 gesendete Nachricht, die vor W1 ankommt (grüne Knoten haben keine weissen Nachbarn!)
  - Wenn ein Knoten grün wird, sind seine Nachbarn bereits vorher rot geworden
  --> Täuschung der Zähler durch Kompensation mittels Nachrichten "rückwärts" über 2 Wellen ist unmöglich!

- *Beweisskizze*:

1) Im roten Gebiet (d.h. zwischen W1 und W2) findet keine Aktivität statt, wenn das Terminierungskriterium S=E=S'=E' gilt:
   Dazu müsste eine vor W1 (im weissen Gebiet) ausgesendete Nachricht im roten Gebiet ankommen. Dann ist aber E' > E.

2) Es kann Nachrichten geben, die beide Wellen "vorwärts" überqueren (d.h. im weissen Gebiet gesendet werden und im grünen ankommen). Solche Nachrichten werden auf S (und S') als gesendet registriert, jedoch weder auf E noch auf E' als empfangen registriert. Da eine Kompensation der Zähler S, E mittels Nachrichten "rückwärts" über 2 Wellen unmöglich ist (wie im Gegenbsp. zum einfachen Zählen), ist dann S > E.

Also: Es gibt keine Nachricht, die W1 überquert; nach W1 findet daher keine Aktivität statt; das System ist nach W1 terminiert

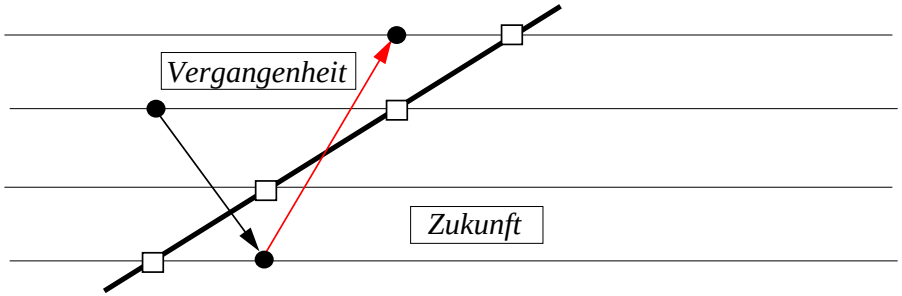
Zeitzonenverfahren

- Erkenntnis: Beim *einfachen Zählen* war eine *irreführende Kompensation* der Zähler durch Nachrichten aus der Zukunft möglich

- Idee: Entlang des Schnittes (induziert durch Welle!) zählen, aber Nachricht aus der Zukunft *erkennen*

- Nachricht aus der Zukunft --> Schnitt inkonsistent

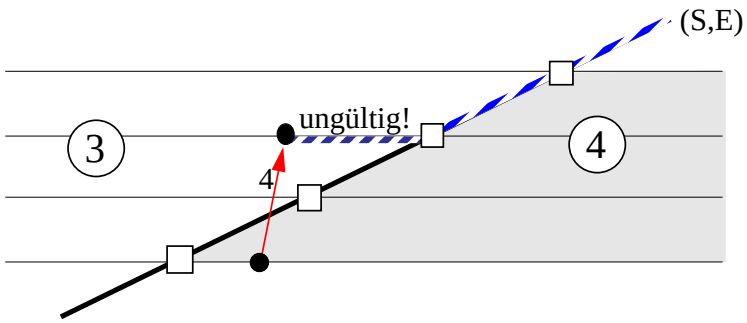
- Keine solche Nachricht --> Schnitt konsistent
  ("äquivalent" zu senkrechtem Schnitt entsprechend Gummibandtransformation)



- Zählergebnis verwerfen, wenn inkonsistent

- Sequenz von Wellen, bis Terminierung festgestellt
  (Liveness ist klar: Schnitt nach Terminierung ist konsistent)

Prinzip: *Erkennen* inkonsistenter Schnitte



Lokale Nachrichtenzähler mit einer Welle akkumulieren, aber Ergebnis *invalidieren*, wenn eine Nachricht aus der Zukunft empfangen wurde

---

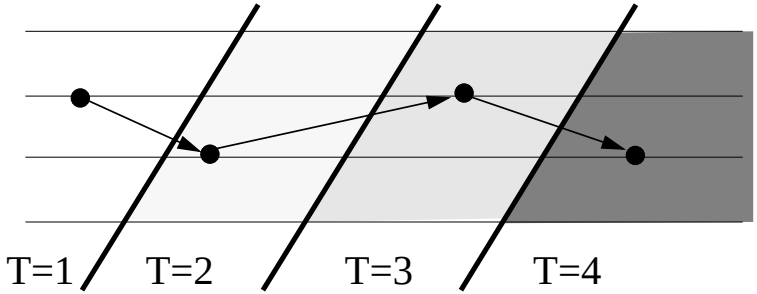
Implementierung:

- "Piggybacking" der Zeitzonennummer auf Nachrichten
- Flag setzen, wenn eine Nachricht aus einer höheren Zeitzone (z.B. "4" im Bild) empfangen wurde
- Welle registriert Flag (und setzt es zurück) und erhöht die Zeitzone

Terminiert, wenn Welle kein gesetztes Flag feststellt und die beiden akkumulierten Zähler übereinstimmen

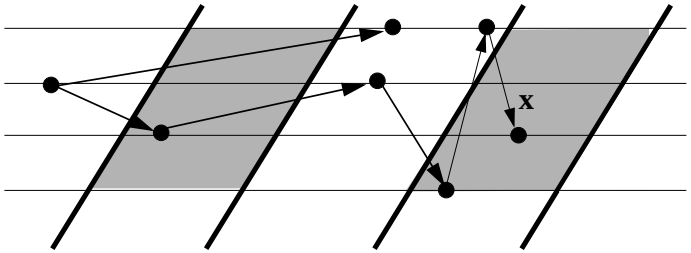
(Formaler Korrektheitsbeweis?)

- Wiederholte Ausführung des zugrundeliegenden Wellenalgorithmus:



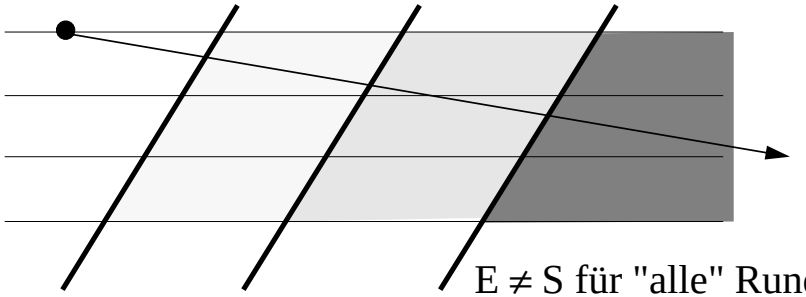
- Zeitzone bei jeder Runde inkrementieren

- "Zyklische" schwarz/weiss-Zeit genügt
  - höhere Zeitzone --> "andere" Zeitzone



- *Jede* Nachricht aus der Zukunft wird erkannt
  - Nachricht aus der Zukunft trägt *andere* Farbe
    (Nachricht rückwärts über zwei oder mehr Wellen existiert nicht)
- Einige Nachrichten aus der Vergangenheit führen zu "Fehlalarmen" (vgl. Nachricht x)
  --> evtl. eine (einzige) zusätzliche Runde nötig

- Anzahl notwendiger Kontrollrunden? Unbeschränkt!



- Vergleich von Doppelzähl- und Zeitzonenverfahren?
  - Aufwand an Kontrollnachrichten und Speicher unwesentlich verschieden
  - Eingriff in die Basisnachrichten bei Zeitzonenverfahren nötig!
  - Zeitzonenverfahren ist nicht "reentrant"
    - lokaler Zustand wird verändert
    - dadurch nicht symmetrisch: Wellen mehrerer Initiatoren ("multi-source") können sich gegenseitig stören

==> Doppelzählverfahren scheint eleganter und einfacher / universeller einsetzbar
(allerdings u.U. eine "Runde" mehr nötig)

- Denkübung: Könnte man nicht Nachrichten aus der Zukunft von vornherein *vermeiden*?
  - z.B. durch Einfrieren des Systems (dann in Ruhe zählen und ggf. in einer weiteren Runde wieder auftauen): ist das korrekt?
  - klappt das Vermeidungsprinzip auch ohne Einfrieren?