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

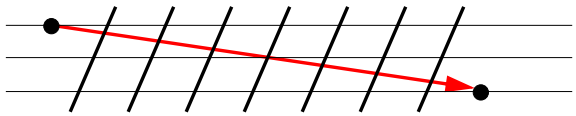
---

"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

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!)

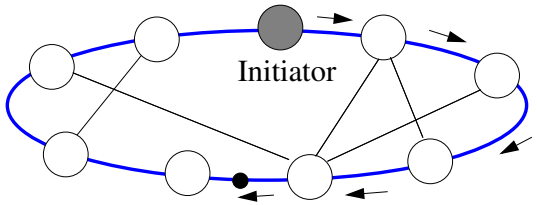
- safe, aber nicht live: meldet stets "weiss nicht"
- live, aber nicht safe: meldet stets "ja"

==> "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

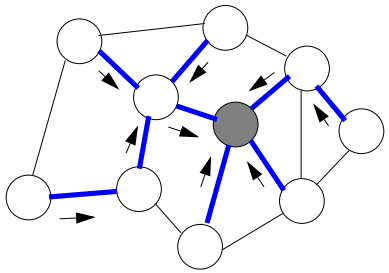
"Logischer" Ring genügt!

- 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



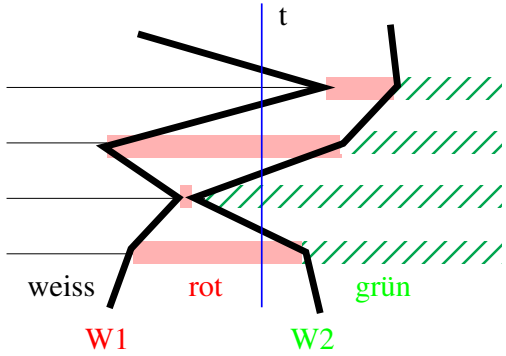
- auch hier werden *zwei* solche zum Initiator hinfliessende Wellen benötigt

- 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!



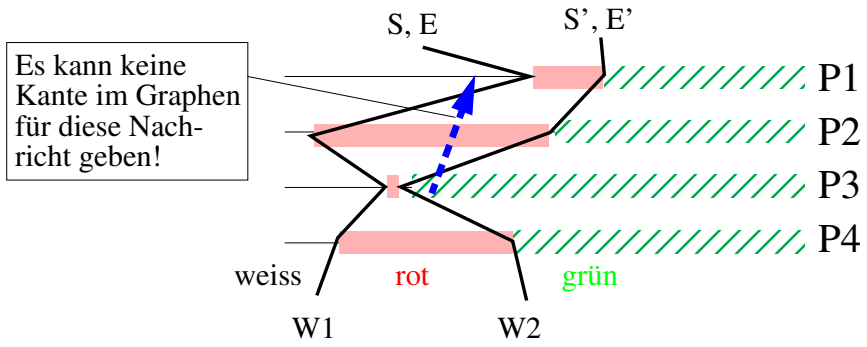
Der Echo-Algorithmus wird als *Transportdienst* zur Realisierung von *zwei* Wellen benutzt, wobei durch die Echo-Nachrichten sowohl die bei W1 lokal gemerkte Information als auch die bzgl. W2 relevante lokale Information an den Initiator übermittelt wird.

- 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

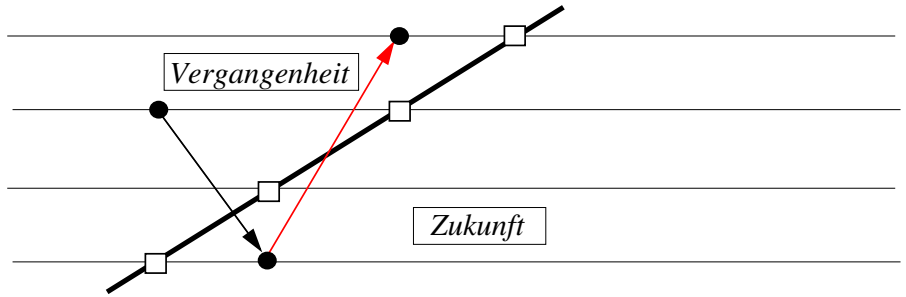
Zeitzonenverfahren

- 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

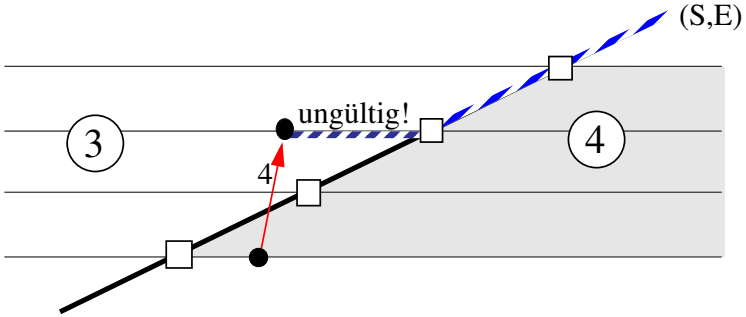
- Keine solche Nachricht --> Schnitt konsistent
  ("äquivalent" zu senkrechtem Schnitt entsprechend Gummibandtransformation)



- 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

---

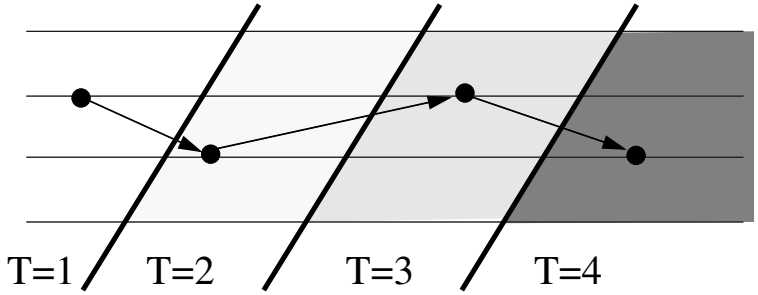
Implementierung:

- "Piggybacking" der Zeitzonennummer auf Nachrichten
- Flag setzen, wenn eine Nachricht aus einer höheren Zeitzone 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

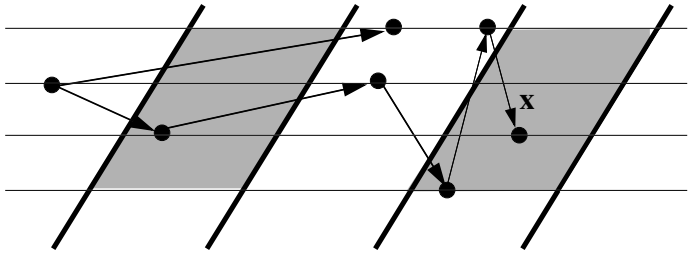
(Formaler Korrektheitsbeweis?)

- Wiederholte Ausführung des zugrundeliegenden Wellenalgorithmus:



- Zeitzone bei jeder Runde inkrementieren

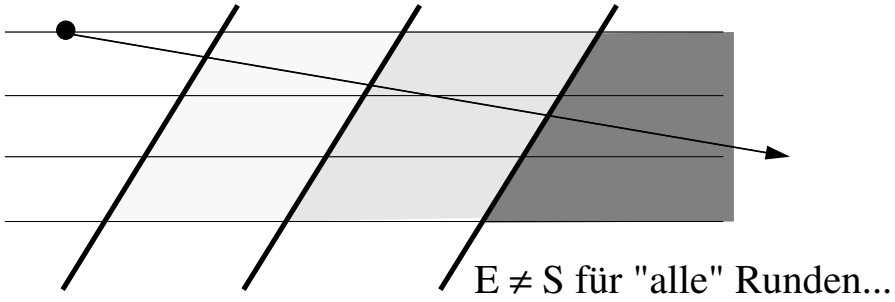
- "Zyklische" schwarz/weiss-Zeit genügt
  - höhere Zeitzone --> "andere" Zeitzone



- *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!



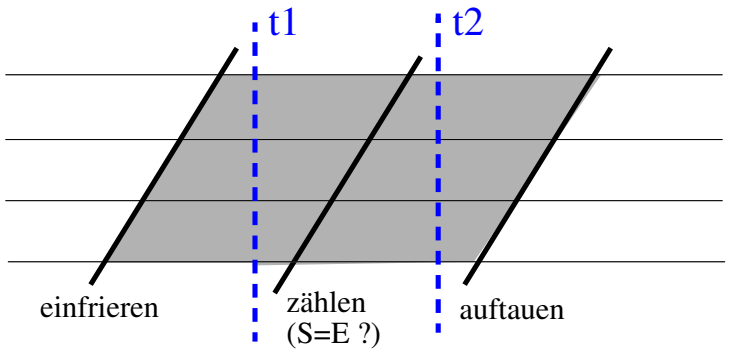
- Vergleich von Doppelzähl- und Zeitzonenverfahren?

  - 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?

Einfrieren?



- "Einfrieren" heisst: Kein Prozess nimmt mehr eine Nachricht an

- Folglich wird im eingefrorenen Gebiet auch keine Nachricht ausgesendet

- Nachrichten, die in diesem Gebiet eigentlich ankommen würden, werden als noch unterwegs befindlich angesehen

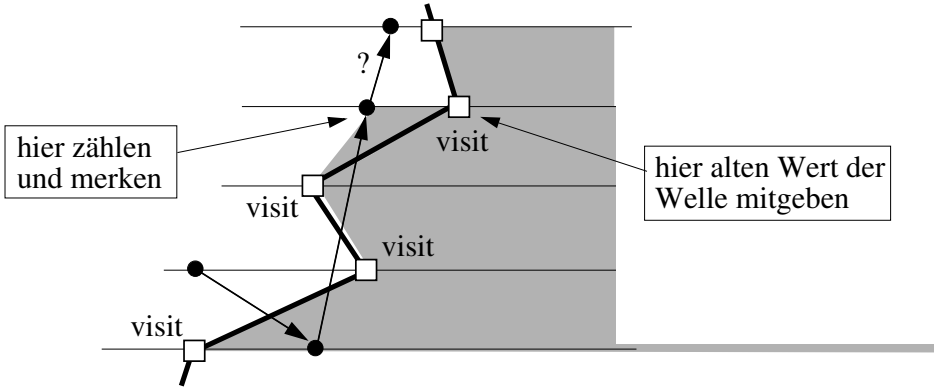
- Man beweise (formal) als Denkübung: Wenn bei der Zählwelle S=E gilt, dann ist die Berechnung terminiert
  - Tip: Man versuche, aus S=E herzuleiten, dass S'=E' bei einem "senkrechten Schnitt" (also z.B. t1 oder t2) gilt

- Informell: Wenn man alles eingefroren hat, dann kann keine Nachricht aus der Zukunft die Zählwelle überqueren

- Wie gut und praktikabel ist dieses Verfahren?

Noch ein anderes Erkennungsprinzip

- Idee: *Vermeiden* inkonsistenter Schnitte durch Vorziehen der Schnittlinie ← d.h. des Kontrollereignisses zum Zählen
  - Prinzip wird uns später beim Schnappschussproblem noch nützlich sein!



- Strategie: Im Augenblick, wo Nachricht aus der Zukunft ankommt, "kurz" vorher das Wesentliche des "visit" Ereignisses ausführen

==> Nachricht verläuft nun ganz in der Zukunft

==> Schnitt ist immer konsistent (--> Zählen ist korrekt)

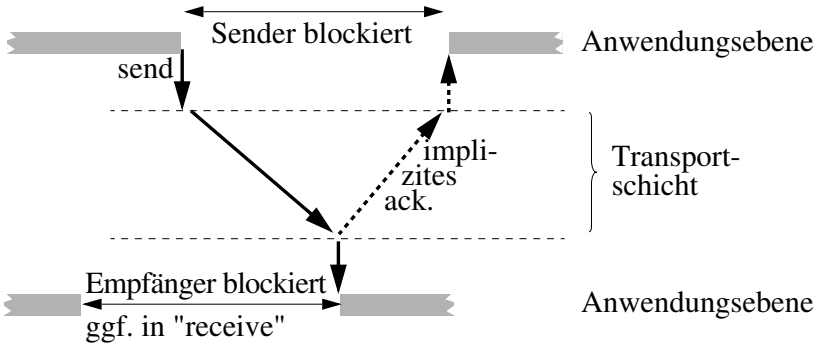
- Bemerkungen:
  - es gibt keinen "Dominoeffekt"; mehrfaches Vorziehen ist nicht möglich
  - formal: Hüllenoperation --> Rechtsabschluss bzgl. der Kausalitätsrelation

- Denkübung: Man vergleiche die bisher vorgestellten Terminierungserkennungsverfahren bzgl. ihrer "Qualität" (=?)
  - z.B.: diese Methode kann im Vergleich zum Zeitzonenverfahren die Terminierung u.U. eine Runde früher feststellen, dafür benötigt sie etwas mehr Speicherplatz (zum "Merken" der Werte)

Synchrones / asynchrones Senden

- *Asynchrones Senden:* Absender wartet nach dem Absenden der Nachricht nicht (“no wait send”)
  --> erfordert ggf. Puffer bei der Realisierung (in der Transportschicht)

- *Synchrones Senden:* Absender wartet blockierend, bis die Nachricht angekommen ist
  --> wartet auf explizite Antwort oder implizites Acknowledgement



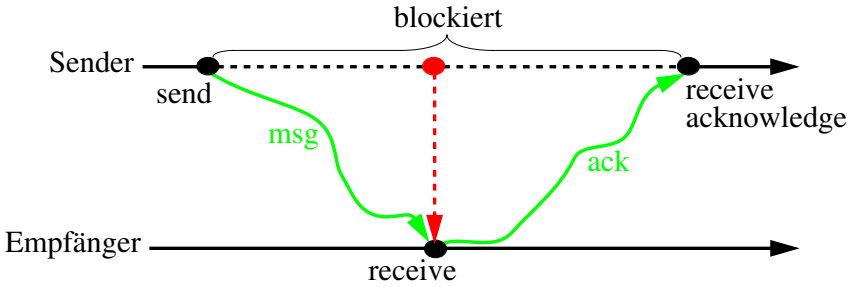
- Aus Sicht des ("bewusstlosen") Senders ist die Nachricht im Augenblick des Sendens auch schon angekommen
  - als wäre die Nachricht "unendlich schnell" (senkrechte Pfeile!)
  - Denkübung: Kann man nicht immer ein Zeitdiagramm per Gummibandtransformation "schadlos" so verzerren, dass ein Nachrichtenpfeil senkrecht verläuft? ← i.w. gleiche Berechnung bei Abstraktion von der Realzeit

---

Achtung: Begriffe synchron / asynchron werden in der Kommunikationswelt verschieden mit leicht unterschiedlichen spezifischen Bedeutungen benutzt!

Synchrone Kommunikation

- Synchrone Kommunikation: syn chron
  "send" und "receive" geschehen virtuell gleichzeitig



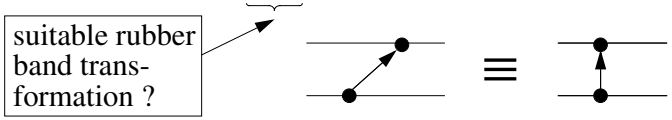
- Sender ist blockiert, bis er vom Empfang seiner Nachricht erfährt
  - Sendezeitpunkt ist innerhalb des Blockadeintervalls beliebig verschiebbar
  - als wäre der Sender vor und nach dem virtuellen Sendezeitpunkt "idle"

- Konsequenz:
  Man darf so tun, als wären Nachrichten nie unterwegs!
  - "blitzschnelle" Nachrichten
  - vereinfacht viele Argumentationen
  - formale Unterscheidung synchron / asynchron durch Kausalrelation (synchron: Abhängigkeit des Senders vom Empfänger)

Modeling Synchronous Communication

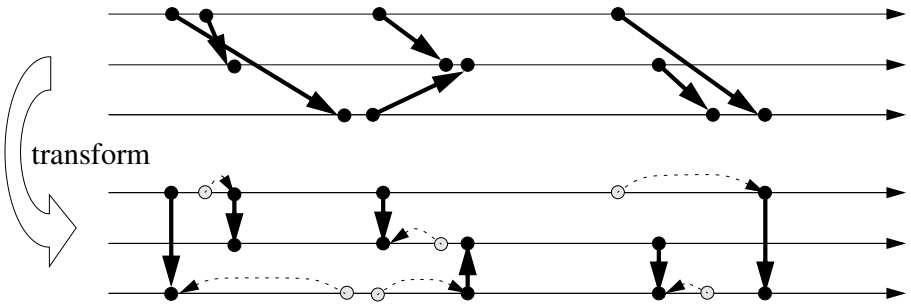
- How do we *define* distributed computations with synchronous message passing?
  - or: *characterize* those distributed computations that can be *realized* with synchronous communications?

- Proposition:
  Synchronous = virtually simultaneous
  = as if msg transmission were instantaneous



- But: aren't instantaneous message transmissions unrealistic?

- Can we *always* apply a suitable rubber band transformation such that all message arrows become vertical?



“As if” Messages were Instantaneous?

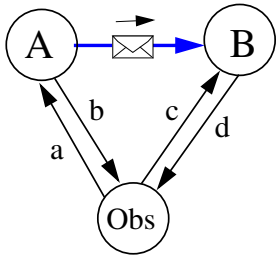
If for a distributed computation a phenomenon can be observed which is impossible with instantaneous messages, the computation must not be realizable with synchronous message passing semantics

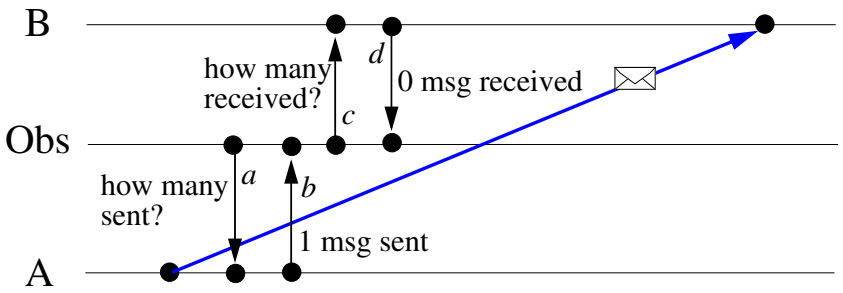
==> message passing should then not be called “synchronous”

Example:

The observer first asks A about the number of messages it sent to B

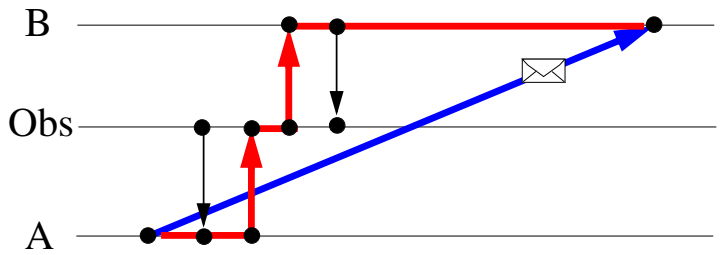
Then it asks B about the number of messages it received from A





Observer learns that a message from A to B is *in transit* for a certain *duration* ==> not synchronous!

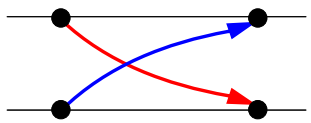
Vertical Message Arrows?



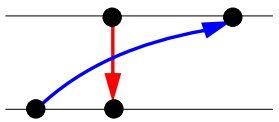
- The message from A to B is *overtaken in an indirect way* by a *chain* of other messages
- The direct message can therefore *not be made vertical* by a rubber band transformation

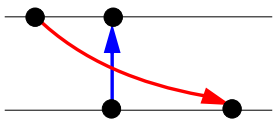
(A message of the chain would then go backwards in time)

- Another computation which is not possible with synchronous communications (==> deadlock):



Although each *single* arrow can be made vertical, it is not possible to draw the diagram in such a way that *both* arrows are vertical!





Besprechung von Übung (1)

Wir wollen hier mündlich und an der Tafel folgende Teilaufgaben aus Übung 1 besprechen, da dies für das Verständnis der folgenden Aspekte ("synchrone Kommunikation") wesentlich ist:

---

g) *Formalisieren* Sie für *Zeitdiagramme* den Begriff (potentiell, indirekt) "*kausal abhängig*" als *Halbordnung* über "*Ereignissen*".

i) Beobachtungen sind eine *lineare Ordnung* von (beobachteten) Ereignissen. In welcher Beziehung steht die oben erwähnte Halbordnung zu dieser linearen Ordnung?