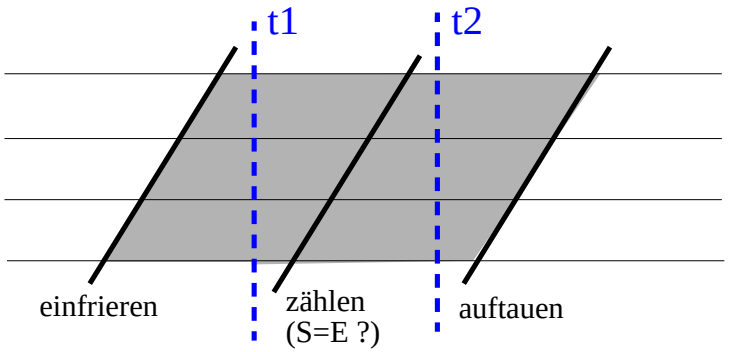
Einfrieren?



- "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

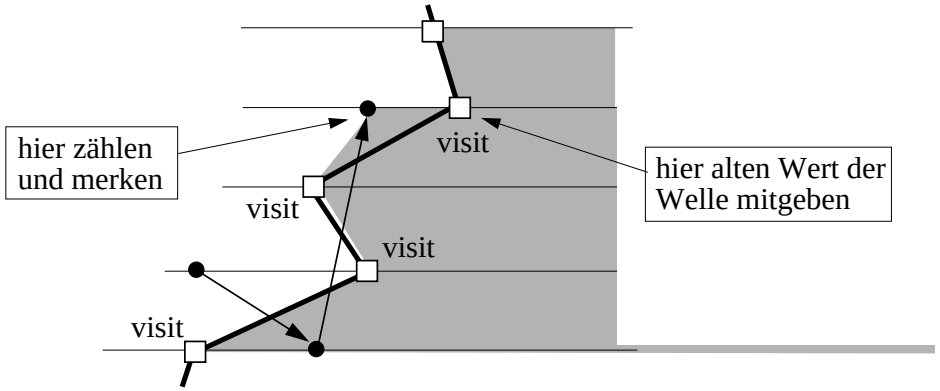
- Informell: Wenn man alles eingefroren hat, dann kann keine Nachricht aus der Zukunft die Zählwelle überqueren

- 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)

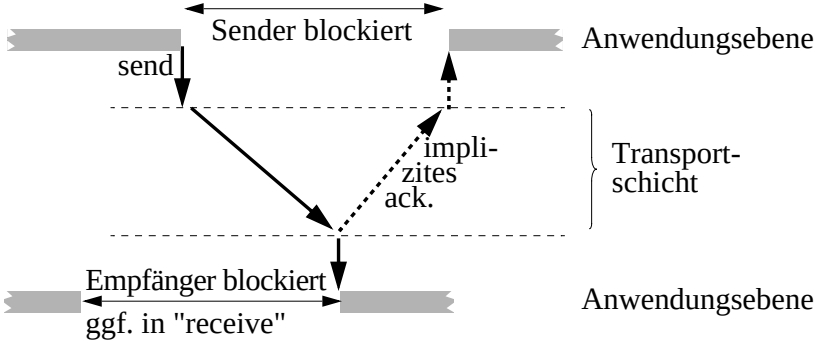
Synchrones / asynchrones Senden

- *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



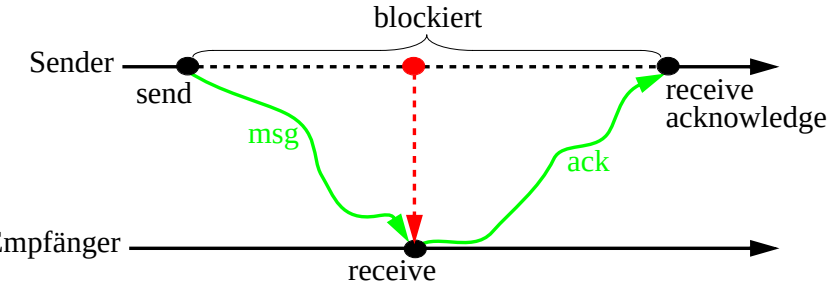
- Aus Sicht des ("bewusstlosen") Senders ist die Nachricht im Augenblick des Sendens auch schon angekommen
  - als wäre die Nachricht "unendlich schnell" (senkrechte Pfeile!)
  - Denkübung: Kann man nicht immer ein Zeitdiagramm per Gummiband-transformation "schadlos" so verzerren, dass ein Nachrichtenpfeil 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”

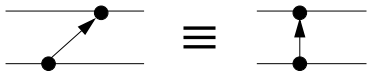
- Konsequenz:
  Man darf so tun, als wären Nachrichten nie unterwegs!
  - “blitzschnelle” Nachrichten
  - vereinfacht viele Argumentationen
  - formale Unterscheidung synchron / asynchron durch Kausalrelation (synchron: Abhängigkeit des Senders vom Empfänger)

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?

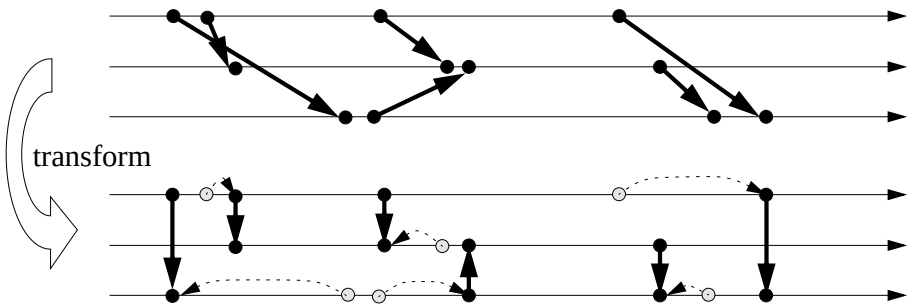
- Proposition:
  Synchronous = virtually simultaneous
  = as if msg transmission were instantaneous

suitable rubber band trans-formation ?



- 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?

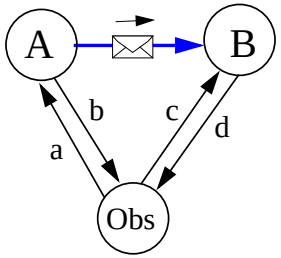
If for a distributed computation a phenomenon can be observed which is impossible with instantaneous messages, the computation must not be realizable with synchronous message passing semantics

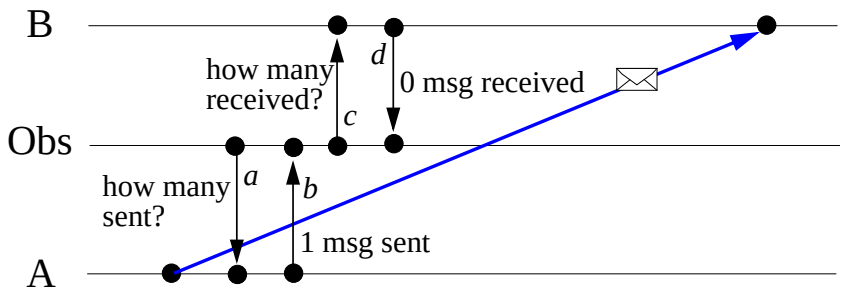
==> message passing should then not be called "synchronous"

Example:

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

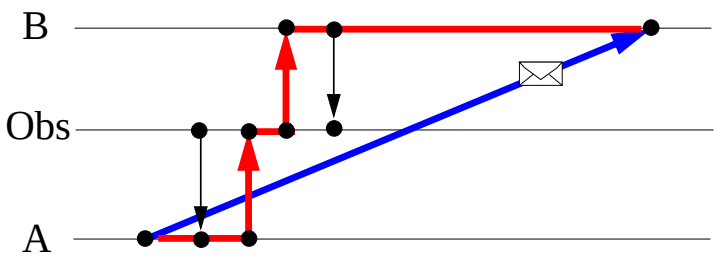
Then it asks B about the number of messages it received from A





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

Vertical Message Arrows?

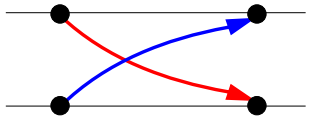


- The message from A to B is *overtaken in an indirect way* by a *chain* of other messages

- 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 an dieser Stelle 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?

Various Characterizations of Synchronous Communications

- Question: are they all equivalent?
- Note: some characterizations are informal or less formal than others

---

1) Best possible approximation of *instantaneous* communications

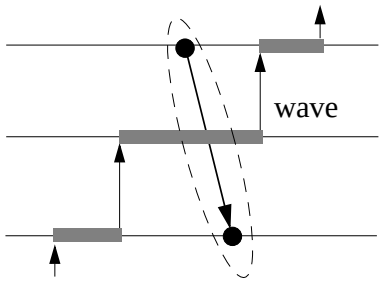
(i.e., without clocks, it is not possible to prove that a message was not transmitted instantaneously)

2) Space-time diagrams can be drawn such that *all message arrows are vertical*

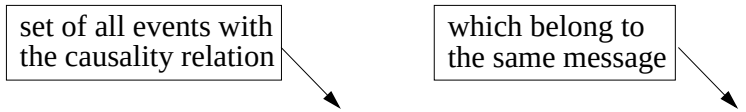
3) *Communication channels* always appear to be *empty*

(i.e., messages are never seen to be in transit)

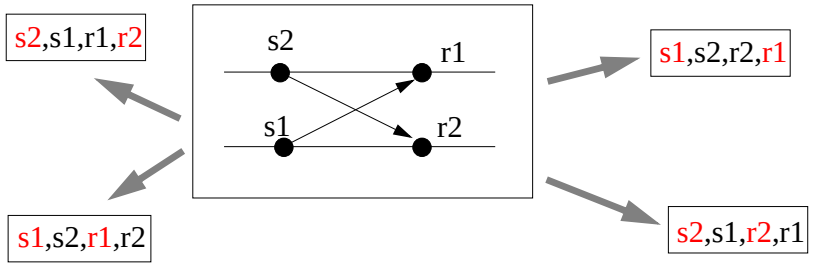
4) Corresponding *send-receive events* form one *single atomic action*



- But what exactly does "atomic" mean?
- Does the combined event happen before or after the wave? Should this be possible with synchronous communication?



5) ∃ *linear extension* of (E, <) such that ∀ corresponding communic. events s,r: *s is an immedate predecessor of r*



- The example has 4 different linearizations: in all of them a pair of corresponding send-receive events is separated by other events - hence this computation cannot be realized synchronously
- Motivation: corresponding events form a single atomic action

6) Define a (transitive) *scheduling relation* '<' on messages:

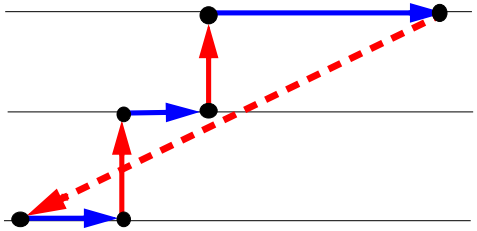
m '<' n iff send(m) < receive(n)

The graph of '<' must be *cycle-free*

this is the classical causality relation on the set of events

- Then whole messages (i.e., corresponding send-receive events s, r) can be scheduled at once (s before r), otherwise this is not possible

7) *No cycle* is possible by moving along *message arrows in either direction*, but always from left to right on process lines



- *Interpretation*: ignoring the direction of message arrows ==>
  - send / receive is "symmetric"
  - "identify" send / receive

- If such a cycle exists ==> no "first" message to schedule
- If no such cycle does exists ==> message schedule exists

8) Synchronous causality relation << is a *partial order*

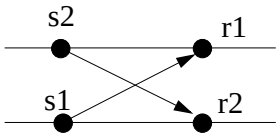
Definition of << : compare this to the classical causality relation!

1. If a before b on the same process, then a << b
2. x << s iff x << r ("common past") } for all corresp. s, r and for all events x
3. s << x iff r << x ("common future") }
4. Transitive closure

Interpretation: corresponding s, r are not related, but with respect to the synchronous causality relation they are "identified"

they have the same past and future

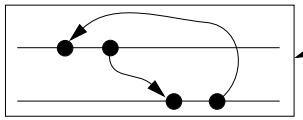
Example:



a) s1 << r2 (1)
b) r1 << r2 (a, 3)
c) s2 << r1 (1)
d) r2 << r1 (c, 3)

→ cycle, but r1 ≠ r2 !

- Compare this characterization to characterization 6
- Why is the definition of << sensible?



This is something where the classical causality relation < is not a partial order, therefore it is not even realizable as a computation with *asynchronous* messages!

Further reading: http://www.inf.ethz.ch/vs/publ/papers/syn_asy.pdf