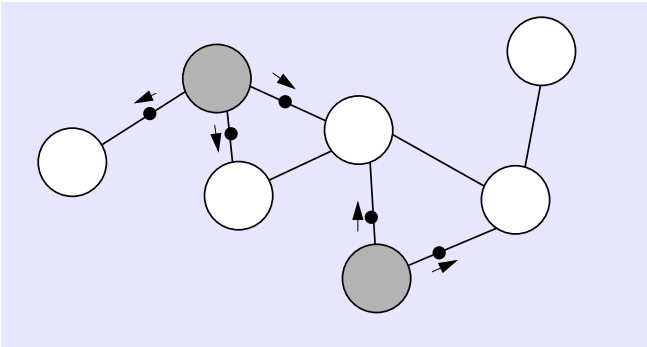
Verteilte Algorithmen

F. Mattern

Departement Informatik
ETH Zürich



Verteilte Algorithmen
= Algorithmen für / in verteilten Systemen

Grundkonzept der Informatik (praktisch / theoretisch)

Interessantes Gebiet der praktischen Informatik sowie wichtiges Anwendungsfeld

- Was ist "verteilt" an einem verteilten Algorithmus?

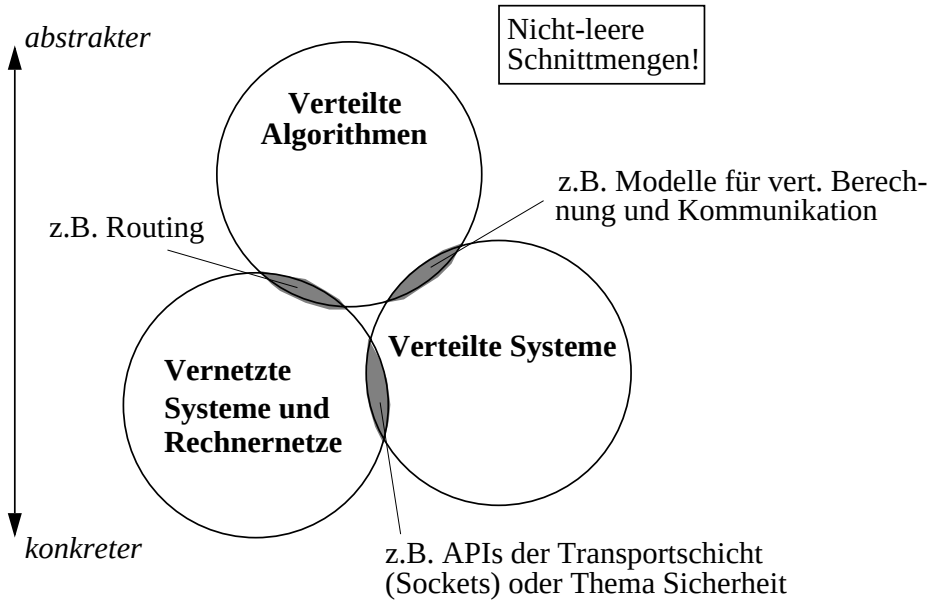
  Vorläufige Antwort: *Zustand* und *Kontrolle* auf verschiedene "Orte"

Einordnung der Vorlesung

- 3-stündige Vorlesung (inkl. Übungen)

- Sinnvolle Vorkenntnisse:
  - Verteilte Systeme (< 10% Überschneidung)
  - Grundkenntnisse der Informatik und Mathematik (Vordiplom)

- Folienkopien jeweils in der darauffolgenden Vorlesung (später auch im WWW im .ps- bzw .pdf-Format)
  - `www.inf.ethz.ch/vs/edu/WS0102/VA/`



Vorlesungsankündigung "Verteilte Algorithmen"

Verteilte Algorithmen sind Verfahren, die dadurch charakterisiert sind, dass mehrere autonome Prozesse gleichzeitig Teile eines gemeinsamen Problems in kooperativer Weise bearbeiten und der dabei erforderliche Informationsaustausch ausschliesslich über Nachrichten erfolgt. Derartige Algorithmen kommen im Rahmen verteilter Systeme zum Einsatz, bei denen kein gemeinsamer Speicher existiert und die Übertragungszeit von Nachrichten i.a. nicht vernachlässigt werden kann. Da dabei kein Prozess eine aktuelle konsistente Sicht des globalen Zustands besitzt, führt dies zu interessanten Problemen.

Im einzelnen werden u.a. folgende **Themen** behandelt:

Modelle verteilter Berechnungen;
Raum-Zeitdiagramme;
virtuelle Zeit; logische Uhren und Kausalität;
Wellenalgorithmen;
verteilte und parallele Graphtraversierung;
Berechnung konsistenter Schnappschüsse;
wechselseitiger Ausschluss;
Election und Symmetriebrechung;
verteilte Terminierung;
Garbage-Collection in verteilten Systemen;
Beobachten verteilter Systeme;
Berechnung globaler Prädikate.

Literatur:

F. Mattern: Verteilte Basisalgorithmen. Springer-Verlag, 1989.
G. Tel: Topics in Distributed Algorithms. Cambridge University Press, 1991.
G. Tel: Introduction to Distributed Algorithms. Cambridge University Press, 2nd edition, 2000.
[ ] V. Barbosa: An Introduction to Distributed Algorithms, MIT Press, 1996.
N. Lynch: Distributed Algorithms, Morgan Kaufmann Pub., 1996.
V. K. Garg: Principles of Distributed Systems, Kluwer, 1996.

Artikel aus Fachzeitschriften (wird in der Vorlesung bekanntgegeben).

Themenspektrum

- Wellenalgorithmen, Traversierungsverfahren
- Verteilte Terminierung
- Verteiltes Garbage-Collection
- Globaler Zustand, Schnappschuss
- Konsistente Beobachtungen und globale Prädikate
- Logische Zeit, Kausalität, Konsistenz
- Wechselseitiger Ausschluss
- Election
- Kommunikationssemantik (synchron, asynchron, kausal)

Algorithmen

- Prinzipielle Phänomene und Begriffe
  - Kausalität, Parallelität,...
- Anwendungsbezug
  - verteilte Betriebssysteme, Datenbanken, Anwendungen, Tools...

- Techniken, Einsichten, Zusammenhänge,...
- Basisverfahren, grundsätzliche Probleme,...
- Abstraktion, Modellbildung, Formalisierung,...
- Problemlösungstechniken, Analysetechniken,...
- Qualitätsbewertung, Komplexitätsabschätzung,...
- Verifikationstechniken

(verallgemeinerbare) Erkenntnisse

Literatur...

Zu einigen Aspekten verteilter Algorithmen, die in dieser Vorlesung angesprochen werden, findet man mehr in **F. Mattern: Verteilte Basisalgorithmen,** Springer-Verlag, IFB 226, 1989. Dort werden auch weitere Literaturhinweise auf spezielle Forschungsliteratur gegeben.

Weitere Aspekte verteilter Algorithmen werden ferner in **G. Tel: Topics in Distributed Algorithms**, Cambridge University Press, 1991 gut, wenn auch etwas formaler und abstrakter, behandelt. 1994 erschien vom gleichen Autor das Buch **Introduction to Distributed Algorithms**, Cambridge University Press, welches eine *empfehlenswerte* und gute Einführung und Übersicht darstellt.

Aus den oben genannten drei Büchern wurden für die vorliegende Vorlesung einige Dinge entnommen. Ferner sei auf weitere Bücher verwiesen:

Das Buch von **V. Barbosa: An Introduction to Distributed Algorithms**, MIT Press, 1996, ist *nicht* empfehlenswert. Das Buch von **N. Lynch: Distributed Algorithms**, Morgan Kaufmann Pub., 1996, behandelt die Problematik umfassend vom theoretischen Standpunkt. Im Buch von **V. K. Garg: Principles of Distributed Systems**, Kluwer, 1996, wird u.a. der Aspekt globaler Prädikate thematisiert. Im Buch von **W. Reisig: Elements of Distributed Algorithms**, Springer-Verlag, 1998, werden verteilte Algorithmen mit Petri-Netzen modelliert und analysiert. Von **M. Raynal** sind einige Bücher in französischer und teilweise auch in englischer Sprache zum Themengebiet "verteilte Algorithmen" erschienen (z.B. "Distributed Algorithms and Protocols, Wiley, 1988"; oder "Networks and Distributed Computation, MIT-Press, 1988"); diese geben jedoch zum grösseren Teil nicht den Stand der Vorlesung wieder.

Leider sind die meisten der in der Vorlesung behandelten Dinge nicht in Buchform, sondern nur in Form von Aufsätzen in verschiedenen Fachzeitschriften veröffentlicht. Die folgende Auswahl führt nur einige wenige auf, die in engerem Bezug zur Vorlesung stehen.

Eine knappe Übersicht zu einigen Themen gibt folgender Aufsatz: "**F. Mattern: Distributed Control Algorithms (Selected Topics)**". In: F. Ozguner (Hg.): Parallel Computing on Distributed Memory Multiprocessors, Springer-Verlag, 1993.

Zum Deadlockproblem stellt "**E. Knapp: Deadlock Detection in Distributed Databases**, Computing Surveys 19:4, 303-328, 1987" eine gute Übersicht dar.

Für das Problem des wechselseitigen Ausschlusses in verteilten Systemen gibt **"B.A. Sanders: The Information Structure of Distributed Mutual Exclusion Algorithms,** ACM Transactions on Computer Systems 5:3, pp. 284-299, 1987" eine interessante Klassifikation.

Das Problem des Garbage-Collections wird allgemein und "klassisch" (also für nicht-verteilte Systeme) im Buch von **Richard Jones, Rafael Lins: Garbage Collection - Algorithms for Automatic Dynamic Memory Management**, Wiley, 1996 dargestellt; zum verteilten Garbage-Collection vgl. man den Aufsatz "**Saleh E. Abdullahi, Graem A. Ringwood: Garbage Collecting the Internet: a Survey of Distributed Garbage Collection**, ACM Computing Surveys, Volume 30, No. 3, Sept. 1998".

... Literatur

Die folgenden Aufsätze behandeln einige Themen, die in der oben als erstes angeführten Monographie nicht oder nur kurz angesprochen werden, genauer:

F. Mattern: Virtual Time and Global States of Distributed Systems. In: Cosnard M. et al. (Hg.): Proc. Workshop on Parallel and Distributed Algorithms, North-Holland / Elsevier, pp. 215-226, 1989.

F. Mattern: Über die relativistische Struktur logischer Zeit in verteilten Systemen. In: J. Buchmann, H. Ganzinger, W.J. Paul (Eds.): Informatik -Festschrift zum 60. Geburtstag von Günter Hotz, Teubner, pp. 309-331, 1992.

G. Tel, F. Mattern: The Derivation of Distributed Termination Detection Algorithms from Garbage Collection Schemes. ACM Transactions on Programming Languages and Systems 15:1, 1-35, 1993.

R. Schwarz, F. Mattern: Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail. Distributed Computing 7:3, 149-174, 1994.

B. Charron-Bost, F. Mattern, G. Tel: Synchronous, Asynchronous, and Causally Ordered Communication. Distributed Computing, 9:4, 173-191, 1996.

F. Mattern et al.: Global Virtual Time Approximation with Distributed Termination Detection Algorithms. Technischer Bericht RUU-CS-91-32, Department of Computer Science, Universität Utrecht, 1991.

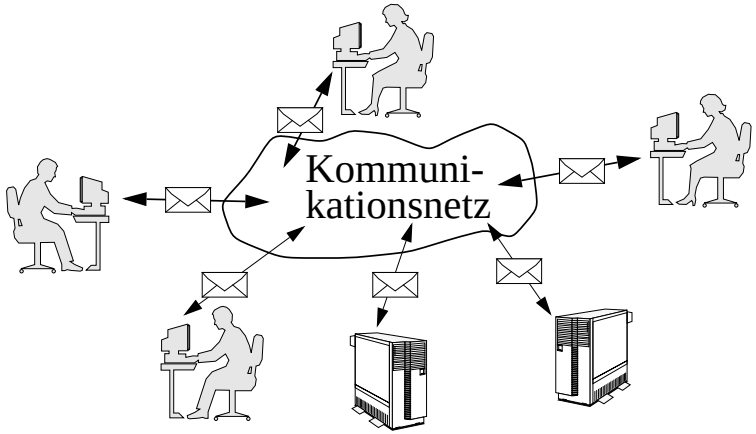
F. Mattern: Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation. Journal of Parallel and Distributed Computing 18:4, pp. 423-434

Die Vorlesung basiert auf einer Vielzahl weiterer Zeitschriftenartikel. Es seien hier nur noch zwei einschlägige aufgeführt:

Der Sammelband "**Global States and Time in Distributed Systems**", herausgegeben von **Z. Yang** und **T.A. Marsland** (IEEE Computer Society Press, 1994), enthält eine grössere Sammlung von Nachdrucken einschlägiger Zeitschriftenartikel der genannten Thematik.

Der Sammelband "**Distributed Systems (second edition)**", herausgegeben von **S. Mullender** (Addison-Wesley, 1993), enthält u.a. den Aufsatz "**Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms**" (pp. 55-96) von **Ö. Babaoglu** und **K. Marzullo**.

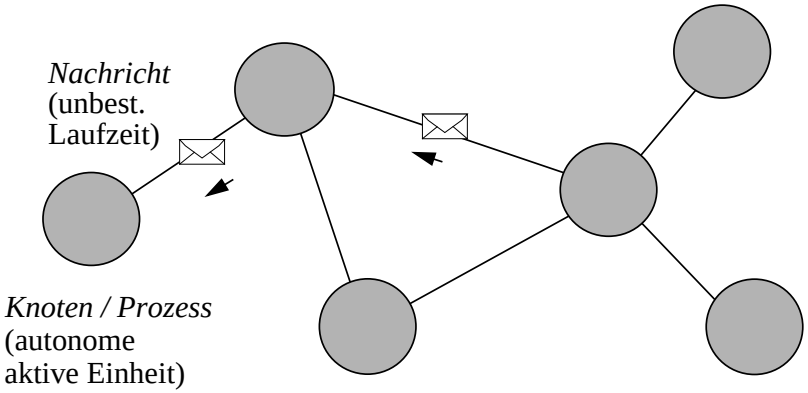
Verteiltes System



- Rechner, Personen, Prozesse, “Agenten” sind an *verschiedenen Orten*

- Autonome Handlungsträger, die jedoch gelegentlich *kooperieren* (und dazu *kommunizieren*)

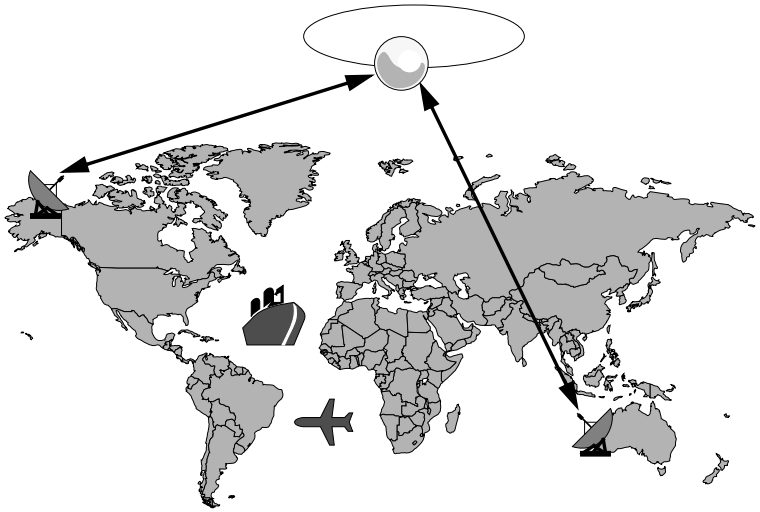
Verteiltes System II



Physisch verteiltes System:
Mehrrechnersystem ... Rechnernetze

Logisch verteiltes System: Prozesse (Objekte, Agenten)
- Verteilung des Zustandes (keine globale Sicht)
- Keine gemeinsame Zeit (globale, genaue "Uhr")

Verteiltes System III

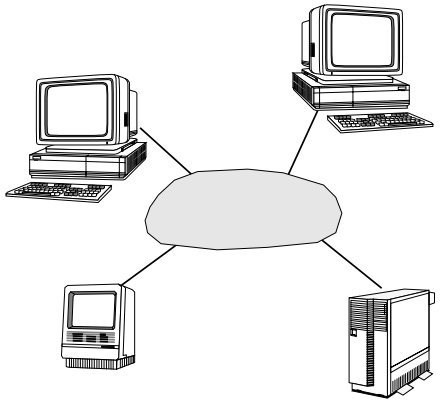


Auch die "reale Welt" ist ein verteiltes System:

- Viele gleichzeitige (“parallele”) Aktivitäten
- Exakte globale Zeit nicht erfahrbar / vorhanden
- Keine konsistente Sicht des Gesamtzustandes
- Kooperation durch explizite Kommunikation
- *Ursache* und *Wirkung* zeitlich (und räumlich) getrennt

- “Inkonsistente Zustände”: Kriegsende zwischen England und Frankreich war in den Kolonialgebieten erst später bekannt!
- Heute: “zeitkompakter Globus” - weitgehend synchronisierte Uhren

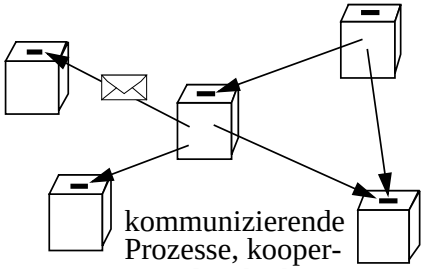
Sichten verteilter Systeme



Rechnernetz mit Rechenknoten:

- Multicomputer (Parallelrechner)
- LAN = Local Area Network
- WAN = Wide Area Network
- Routing, Adressierung....

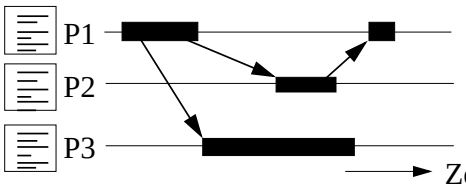
zunehmende Abstraktion



kommunizierende Prozesse, kooperierende Objekte

Objekte eines *Betriebssystems* bzw. einer *Programmiersprache*

==> “Programmierersicht” (Client, Server...)



Algorithmen- und Protokoll-ebene

- Aktionen, Ereignisfolgen
- Konsistenz, Korrektheit

Sichten (2)

- Von Technologie bis zu Prinzipien
  - physische Sicht: z.B. Techniker
  - Programmiersicht: z.B. Programmierer, Systementwickler
  - abstrakte Sicht: z.B. Algorithmentheoretiker

- Entspricht dem Kanon verschiedener Lehrveranstaltungen:
  - „Rechnernetze“: technische Eigenschaften, Topologien, Realisierungsweisen
  - „Verteilte Systeme“: Infrastruktur, Komponenten, Protokolle
  - „Verteilte Algorithmen“: Korrektheit, grundsätzliche Phänomene, Konzeptualisierung

- „*Logisch* verteilte Systeme“: unabhängig von *physischer* Verteilung
  - fehlender globaler Zustand, fehlende globale Zeit
  - Parallelität
  - Autonomie (z.B. unabhängige Prozesse, Objekte, Agenten …)

Parallele <--> verteilte Algorithmen

Sequentieller Algorithmus

↓

Paralleler Algorithmus

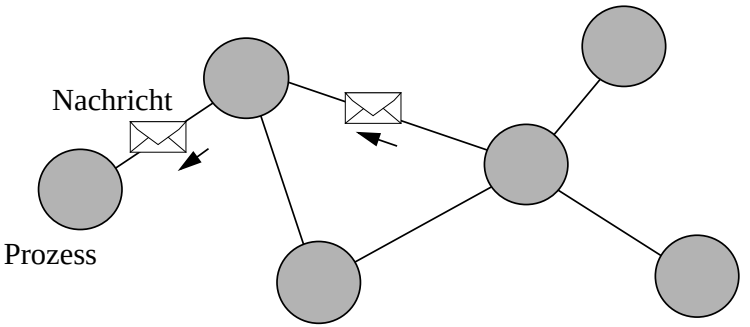
- mehrere, oft gleichartige Prozesse
- Synchronisation oft über gemeinsamen Speicher
- Zweck: Beschleunigung gegenüber seq. Algorithmus

↓

Verteilter Algorithmus

- mehrere Prozesse kommunizieren über Nachrichten
- gemeinsames Ziel --> Kooperation
- ideal: keine zentrale Kontrolle

oft asynchron



Aspekte verteilter Systeme

im Vergleich zu *sequentiellen* Systemen:

- Grösse und Komplexität → jede(r) ist anders
- Heterogenität → jede(r) ist anders
- Nebenläufigkeit → vieles gleichzeitig
- Nichtdeterminismus → morgen anders als heute...
- Zustandsverteilung → niemand weiss alles

- Programmierung *komplexer*
- Test und Verifikation *aufwendiger*
- Verständnis der Phänomene *schwieriger*

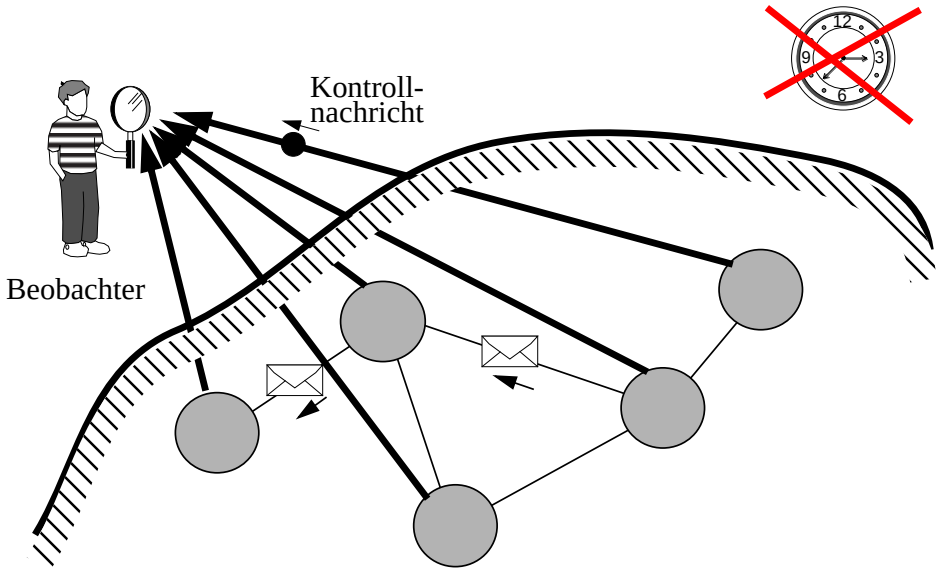
==> gute Werkzeuge (“Tools”) und Methoden
- z.B. Middleware als Software-Infrastruktur

==> adäquate Modelle, Algorithmen, Konzepte
- zur Beherrschung der neuen Phänomene

Ziel: Verständnis der grundlegenden Phänomene, Kenntnis der geeigneten Konzepte und Verfahren

Phänomene verteilter Berechnungen

Beobachten verteilter Berechnungen



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

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

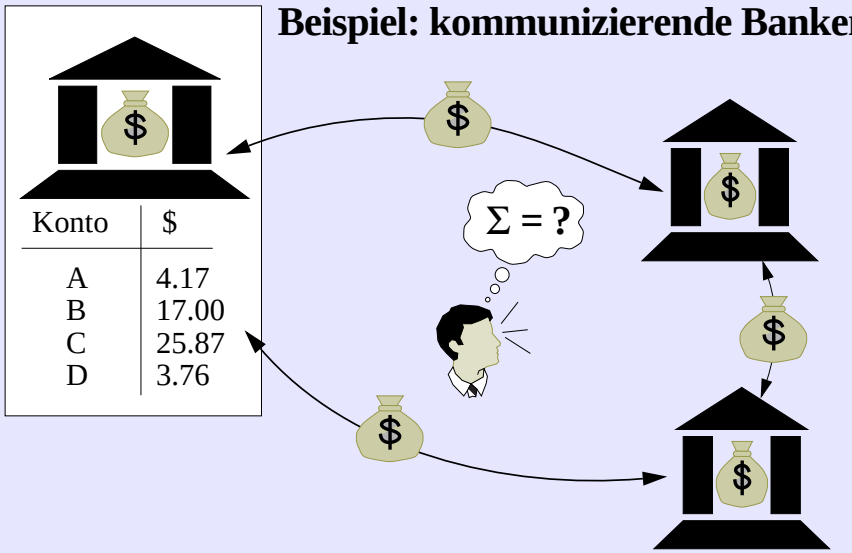
was heisst das?

"Korollar": Aussagen über den globalen Zustand sind schwierig

Ein erstes Beispiel: Wieviel Geld ist in Umlauf?

- konstante Geldmenge, oder
- monotone Inflation (--> Untergrenze)

Beispiel: kommunizierende Banken



- Modellierung:
  - verteilte Geldkonten
  - ständige Transfers zwischen den Konten
- Erschwerte Bedingungen:
  - niemand hat eine globale Sicht
  - es gibt keine gemeinsame Zeit (“Stichtag”)

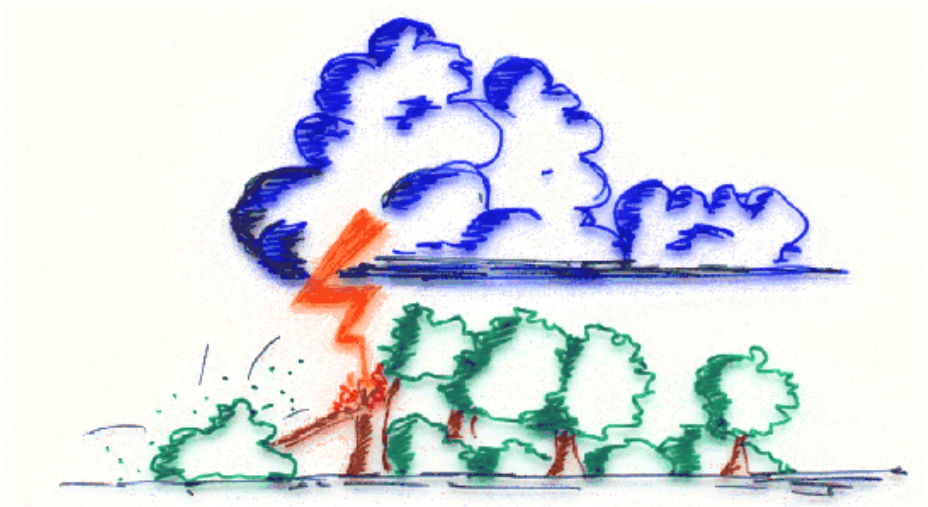
--> Geht das dann überhaupt?

--> Ist das überhaupt ein wichtiges Problem?

==> *Schnappschussproblem*

Beispiel: Prähistorische Gesellschaft

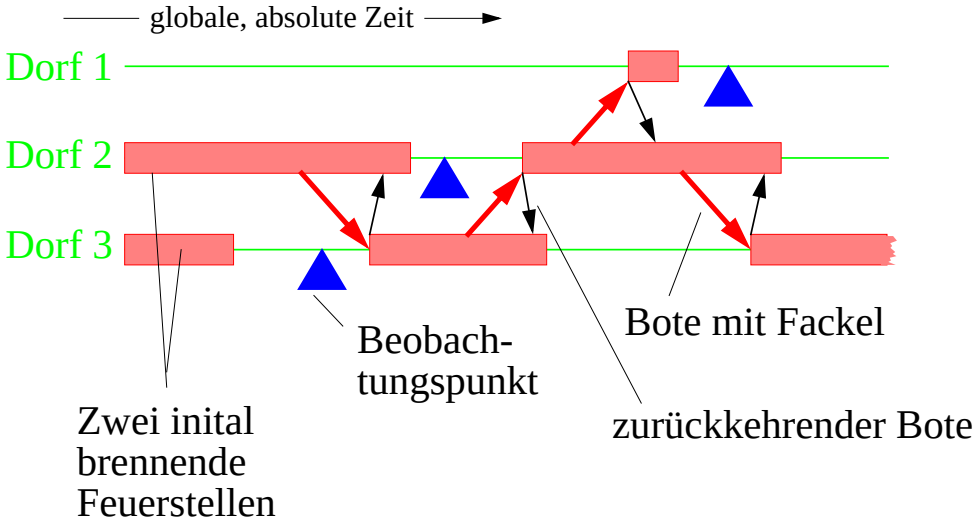
- Einzelne, versprengte Stämme
- Beschränktes technisches Wissen
  - Feuermachen unbekannt
  - Feuerhüter ist ein angesehener Beruf!
- Feuer erlischt--> von einem anderen Stamm holen
- Nur lokale Sicht
  - Glimmt noch ein Fünkchen Hoffnung?
  - Oder ist der Ofen aus?
- Alle Feuer erloschen und kein Feuerträger unterwegs:
  - Warten auf einen Blitz (--> grosser Feuerzauber...)



- Feststellen der Terminierung ist wichtig
  - kein warmes Essen bis zum nächsten Gewitter...

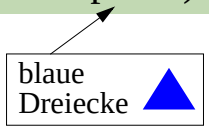
Falsche Beobachtungen

Beobachter kann nicht alles gleichzeitig betrachten!



Für alle Feuerstellen gilt (zu einem Zeitpunkt):

- kein Feuer brennt
- kein Bote ist unterwegs

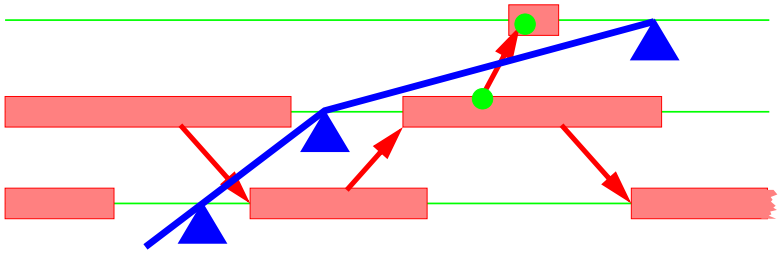


Aber: Es gibt keinen *einzigen Zeitpunkt*, wo kein Feuer brennt!

==> Beobachtung liefert ein *falsches ("schiefes") Bild!*

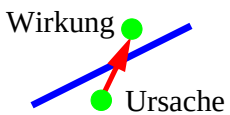
==> *Terminierungserkennungs-Problem*

Was läuft schief?



- Es scheint, als ob alle Feuer aus sind und genauso-viele Boten angekommen wie aufgebrochen sind
  - dann wäre tatsächlich “alles aus” (bis zum nächsten Blitzschlag)

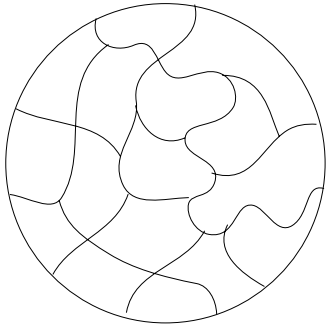
- Täuschungsgrund: Beobachtung ist nicht kausaltreu



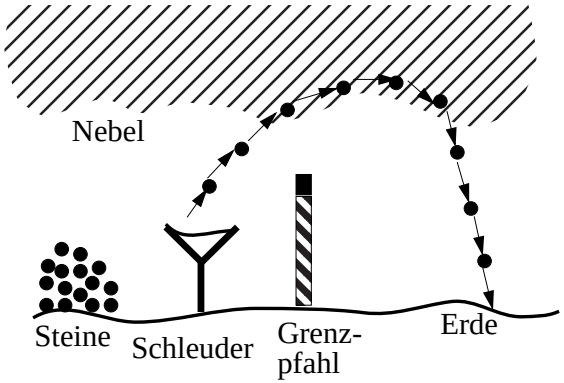
  - Wirkung wird registriert, nicht aber die zugehörige Ursache
  - inkonsistente Sicht durch schiefen Schnitt

- Würde man alles gleichzeitig beobachten, dann könnte man niemals eine Wirkung ohne ihre Urssache wahrnehmen!

Wie stellt man fest, dass der ewige Friede ausgebrochen ist?



Die Welt ist vollständig in einzelne Gebiete parzelliert



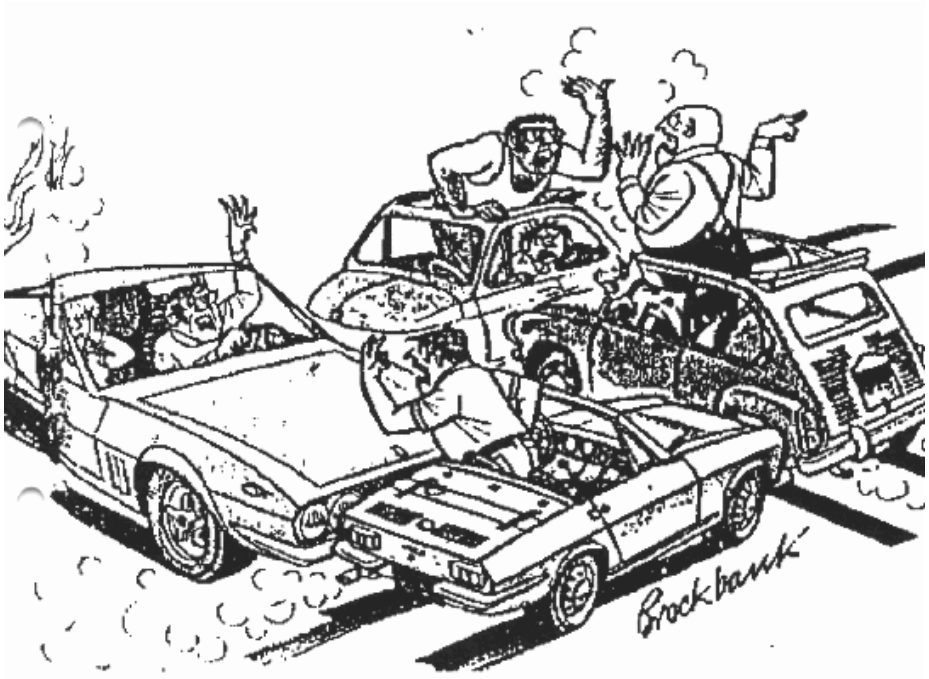
Regel: Nur wer von einem Stein getroffen wird, *darf* Steine wegschleudern

- Urknall: Ein einziger Meteorit traf die Erde...
- Steine fliegen beliebig hoch und brauchen beliebig lang...
- Niemand hat den Gesamtüberblick...

- Der Friede ist ewig ("stabil"; "monotone Eigenschaft")
- Wie stellt man sicher fest, *dass* er gilt (*wenn* er gilt)?

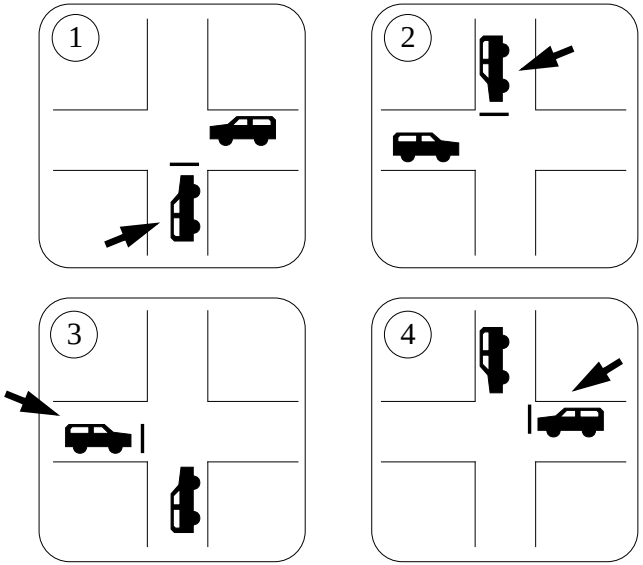
==> *Terminierungserkennungs-Problem*

Das Deadlock-Problem



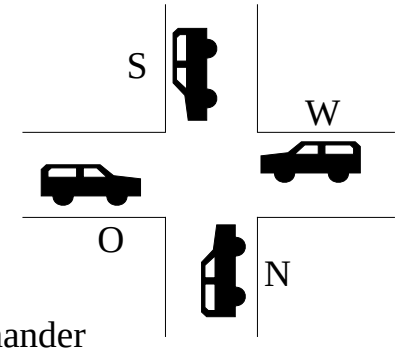
„Neapolitanischer Hakenkreuzstau“
(Also sprach Bellavista, Luciano De Crescenzo)

Phantom-Deadlocks



Vier Einzelbeobachtungen der Autos N, S, O, W

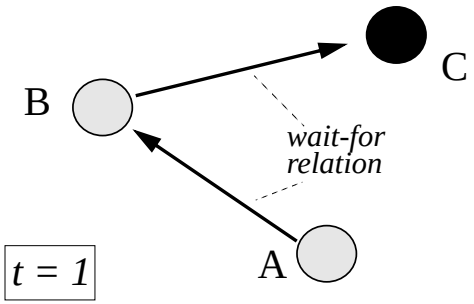
1) N wartet auf W
2) S wartet auf O
3) O wartet auf N
4) W wartet auf S



zu notwendigerweise verschiedenen Zeitpunkten liefert den *falschen* Eindruck, als würden zu einem *einzigen* Zeitpunkt alle zyklisch aufeinander warten (--> Verklemmung)

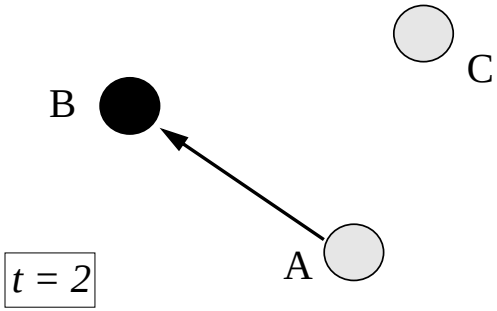
==> *verteiltes Deadlockproblem*

Phantom-Deadlocks

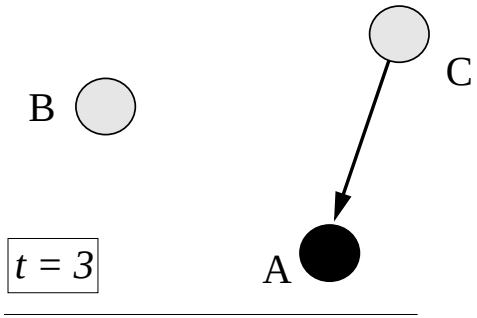


(C benutzt ein exklusives Betriebsmittel)

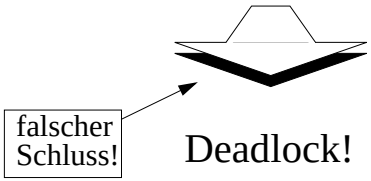
beobachte B:
==> B wartet auf C



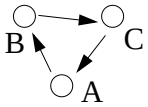
beobachte A:
==> A wartet auf B



beobachte C:
==> C wartet auf A

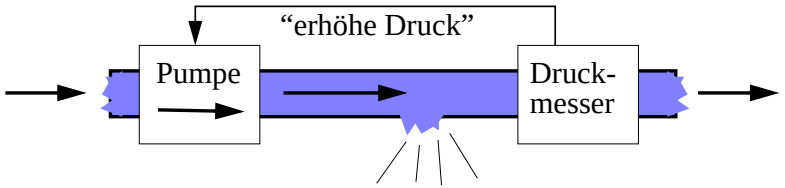


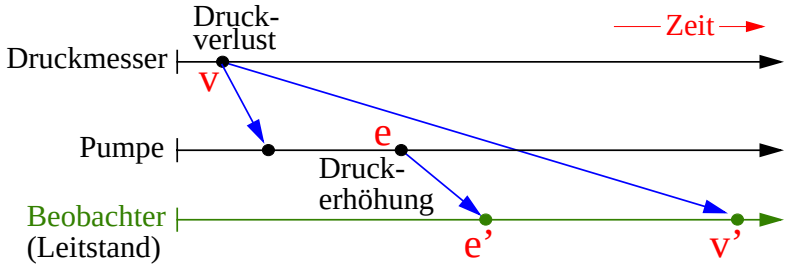
Deadlock!



Kausal (in)konsistente Beobachtungen

- Gewünscht: Eine Ursache stets vor ihrer (u.U. indirekten) Wirkung beobachten



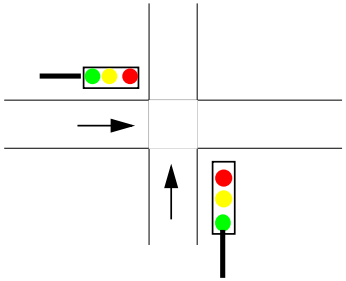


Falsche Schlussfolgerung des Beobachters:
Es erhöhte sich der Druck (aufgrund einer unbegründeten Aktivität der Pumpe), es kam zu einem Leck, was durch den abfallenden Druck angezeigt wird.

==> *"Causal order-Problem"*

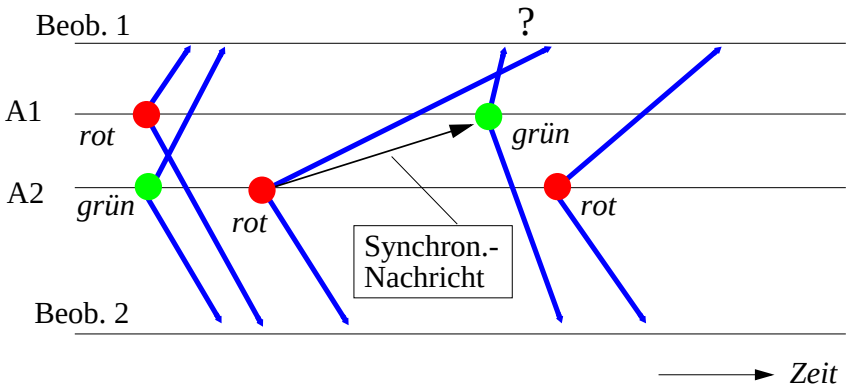
Beispiel: Verteilte Ampelsteuerung

(hier für 2 Ampeln)



Konsistenzproblem bei *mehreren Beobachtern*

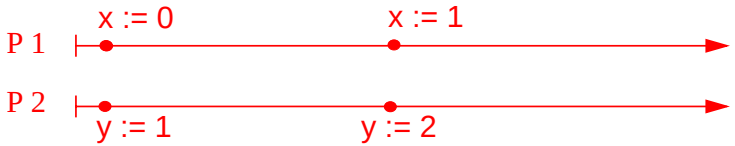
- Jede Ampel darf autonom auf rot schalten
- Eine Ampel darf nur dann auf grün schalten, wenn sie erfahren hat, dass die andere rot ist
- Umschalten der Ampel ist ein *Ereignis* ●
  (*atomar*: zeitlos, nicht unterbrechbare Aktion)

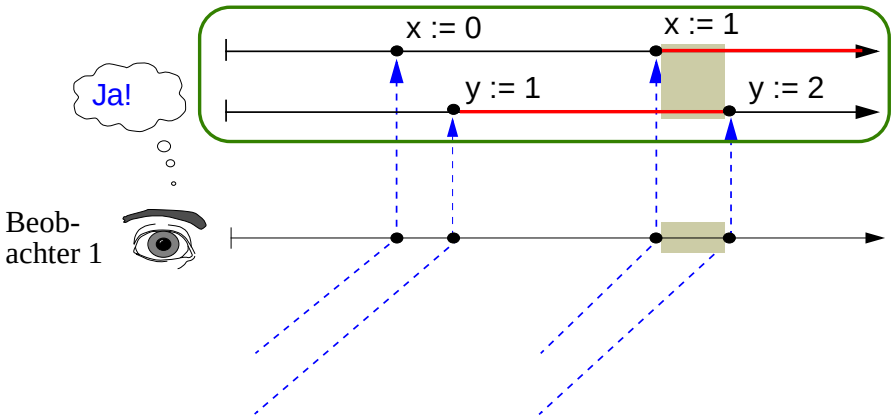


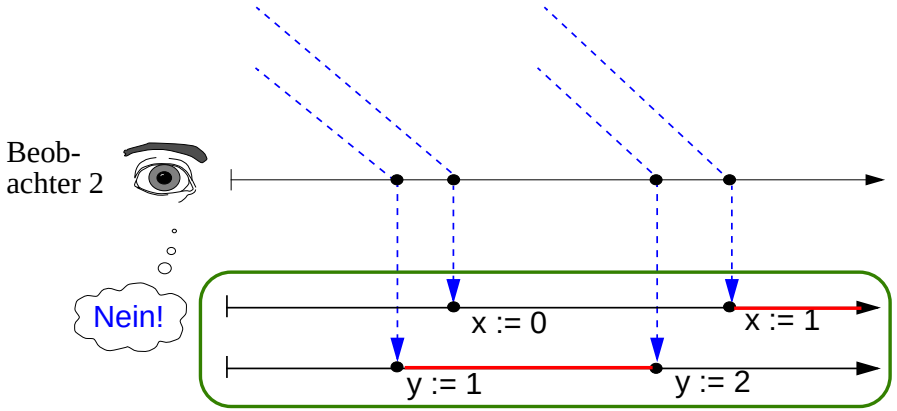
- Welcher Beobachter hat Recht? (Wieso?)
- Ähnliche Probleme (in komplexerem Umfeld) tauchen in der Praxis tatsächlich auf (z.B. Flugkontrollsystem)

Das Problem globaler Prädikate

Frage: Gilt in der vorliegenden Berechnung x = y ?





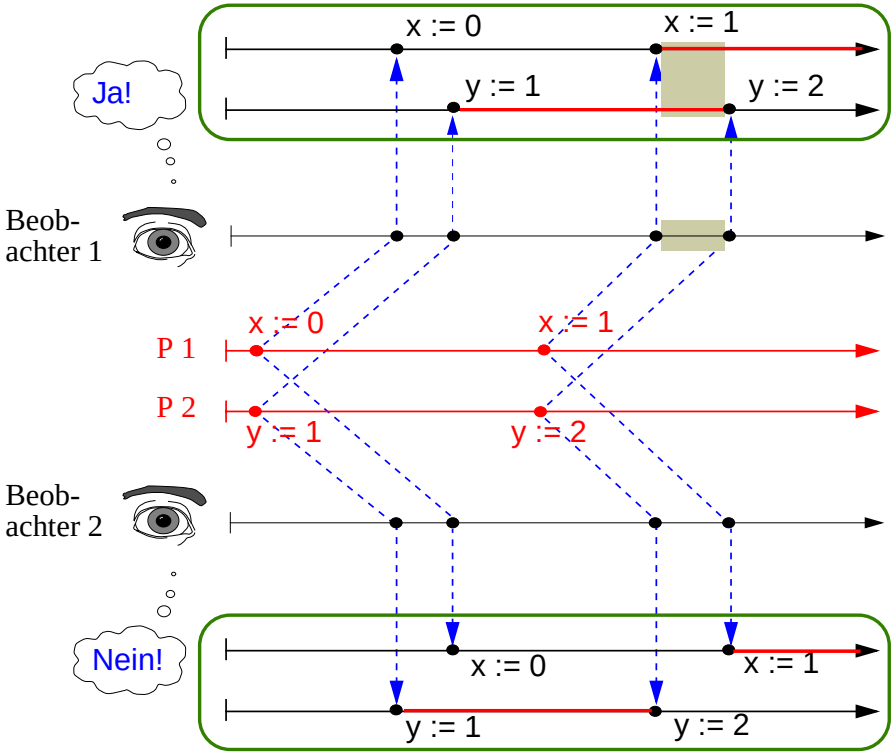


- Beide Beobachtungen sind gleich “richtig”
- Die Beobachter stimmen bzgl. x = y nicht überein!

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

Das Problem globaler Prädikate

Frage: Gilt in der vorliegenden Berechnung x = y ?



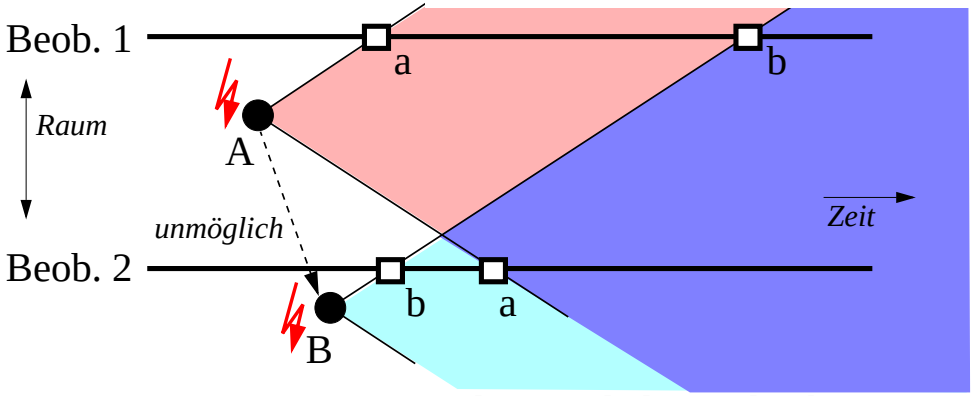
- Beide Beobachtungen sind gleich “richtig”
- Die Beobachter stimmen bzgl. x = y nicht überein!

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

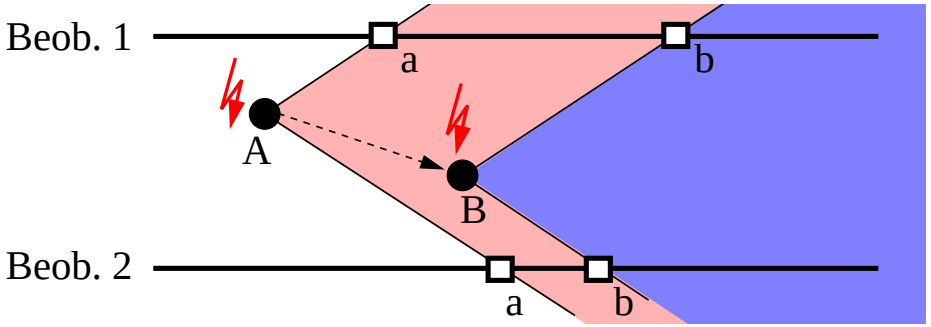
Relativierung der Gleichzeitigkeit

Zwei "kausal unabhängige" Ereigniss können in beliebiger Reihenfolge beobachtet werden!

Lichtkegel-Prinzip der relativistischen Physik



A und B geschehen "gleichzeitig"
(jedenfalls *kausal unabhängig*!)

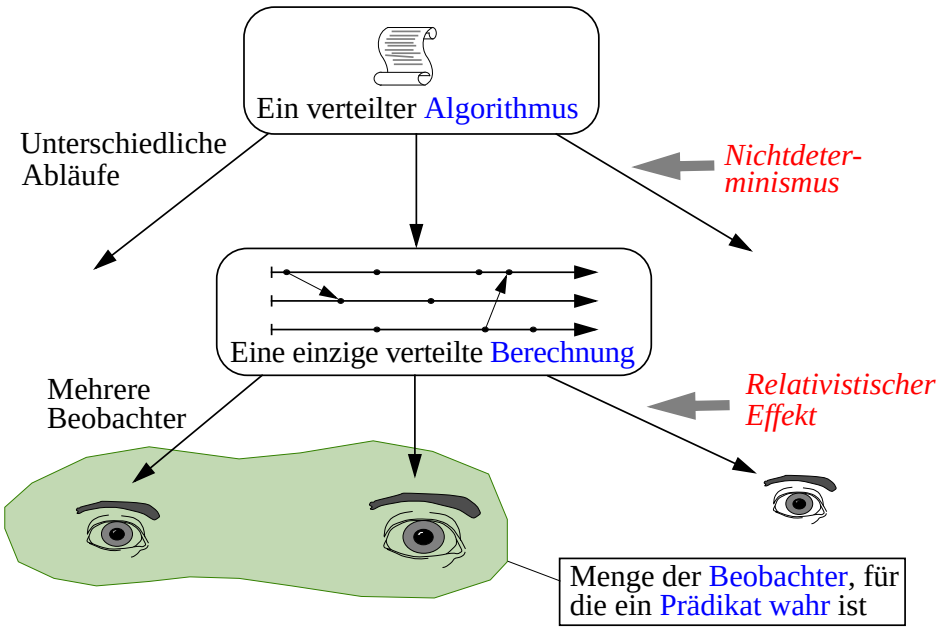


beobachterinvariant
==> *objektive Tatsache*

B liegt im Kegel von A -->
B hängt kausal ab von A -->
Alle Beobachter sehen B nach A

Die "Beobachtungsvielfalt"

- Verschiedene Beobachter sehen *verschiedene Wirklichkeiten*



Konsequenz:

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

Ursache:

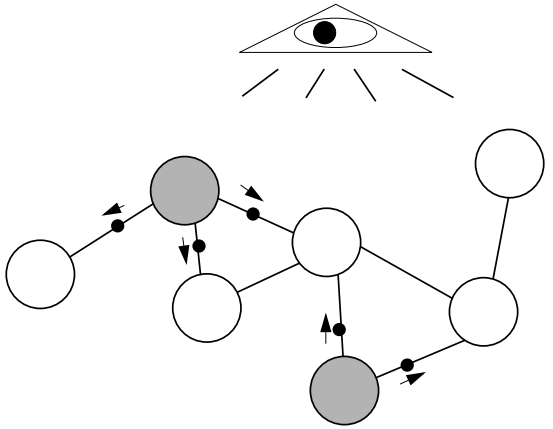
Bei sequentiellen Berechnungen fallen Berechnung und Beobachtung zusammen, im verteilten Fall nicht!

Halbordnung / lineare Ordnung

Globale Prädikate...

Eigenschaften globaler Zustände

... sind aber wichtig!



Bsp.:

- Wieviel Geld ist in Umlauf?
- Ist die parallele Approximation gut genug?
- Ist die verteilte Berechnung terminiert?
- Ist ein bestimmtes Objekt "Garbage"?
- Konsistenter Sicherungspunkt (vert. Datenbank)?
- Wie hoch ist die momentane Last?
- Liegt eine zyklische Wartebedingung vor?

Einige globale Prädikate lassen sich "eindeutig" bestimmen

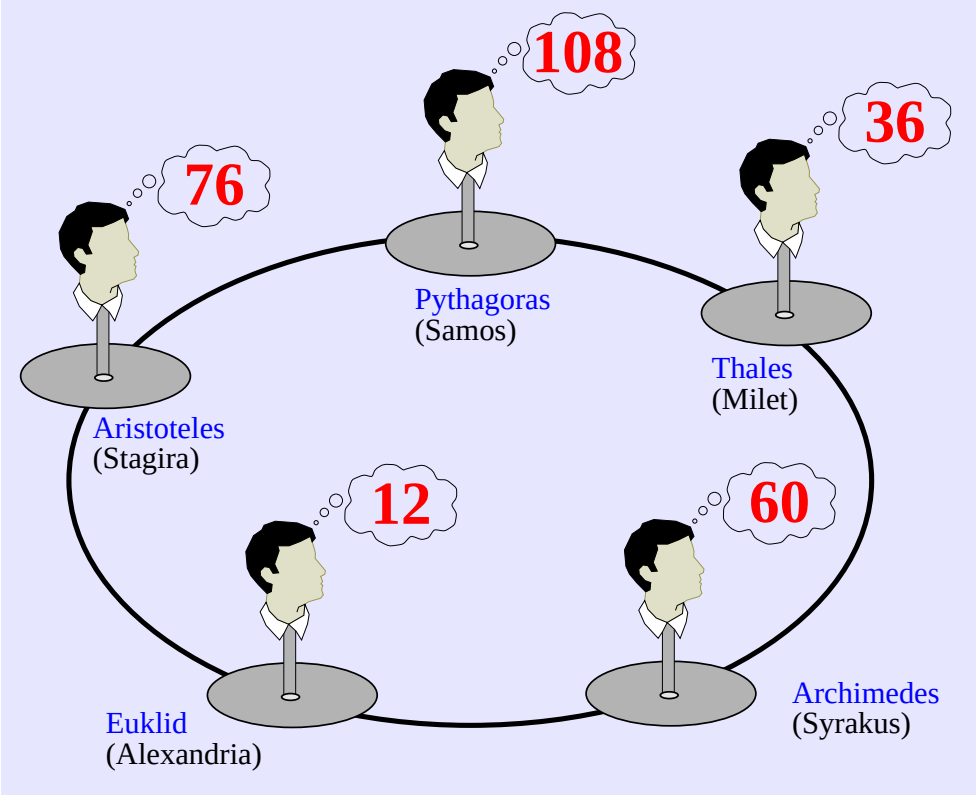
- Welche? --> fundamentale Aspekte analysieren
- Wie? --> *Algorithmen*

u.a. Thema der Vorlesung

Erste Beispiele für verteilte Algorithmen

Ein erster verteilter Algorithmus: Verteilte Berechnung des ggT

Fünf in Raum und Zeit verteilte griechische Philosophen wollen zusammen den grössten gemeinsamen Teiler ihres Alters berechnen...



- Was ist der ggT einer Menge $\{a_1,...,a_n\}$ nat. Zahlen?
- Wie funktioniert der *euklidische Algorithmus*?

"Elemente" Buch 7, Satz 1 und 2

Der grösste gemeinsame Teiler (ggT)

- Nachfolgend werden nur natürliche Zahlen betrachtet
- t heisst *Teiler* einer Zahl p, falls es q ≠ 0 gibt mit q t = p
  - Bsp: 6 ist Teiler von 30 (da $5 \times 6 = 30$)
    3 ist Teiler von 30 (da $10 \times 3 = 30$)
    aber: 8 ist kein Teiler von 30
- 1 ist Teiler jeder Zahl (klar nach Definition)

- t heisst *gemeinsamer Teiler* von u, v
  wenn t Teiler von u ist und t Teiler von v ist
  - Bsp: 6 ist gemeinsamer Teiler von 30 und 18.
    Aber 3, 2, 1 sind auch gemeinsamer Teiler.
    Jedoch gibt es keine anderen gemeinsamer Teiler
    ==> 6 ist der grösste gemeinsame Teiler (ggT).
- Zu je zwei Zahlen ≠ 0 gibt es stets einen ggT, da die Menge der gemeinsamen Teiler mindestens die 1 enthält und endlich ist (was aber ist ggT(x,0)?)
- Kanonische Erweiterung auf n ≥ 1 Argumente
  - Bsp: ggT(108, 36, 60, 12, 76) = 4

- *Anwendung*: Z.B. Kürzen von Brüchen

Satz von Euklid

Der ggT zweier positiver ganzer Zahlen x,y (mit x≥y>0) ist gleich dem ggT von y und dem Rest, der bei ganzzahliger Division von x durch y entsteht

- Offenbar ist ggT(x,x) = x für alle x
- Man setzt nun noch ggT(x,0) = x für alle x
- Obiger Satz legt eine rekursive Realisierung nahe

```
ggt(x,y) --> ggt(y,mod(x,y)
```

  - für x > y werden in der Rekursion beide Argumente jeweils kleiner
  - Rekursion geht irgendwann mit ggT(...,0) zuende
  - oft: iterative Formulierung (statt rekursiver)

==> *Euklidischer Algorithmus*

Vorteil: Zur Bestimmung des ggT sind Primfaktorzerlegung ("Schulmethode") oder Probedivisionen nicht notwendig

Hinweis: Der Beweis des Algorithmus (und des Satzes) ist eine interessante Wiederholungsübung!

Das (abstrakte) Lösungsprinzip

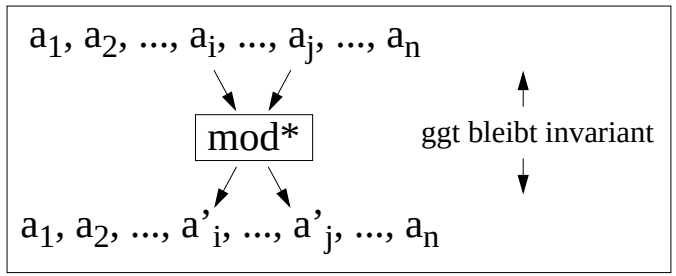
- *Invariante* (für $x \geq y > 0$):

  $ggT(x,y) = ggT(y,mod^*(x,y))$

  Wie *Restfunktion* mod, soll jedoch y liefern, falls x ganzzahliges Vielfaches von y ist. Also: mod*(a,b) = mod(a-1,b)+1

- *Idee:* Ersetze x durch mod*(x,y) ← Ist i.a. kleiner als x

- Erweiterung auf n Zahlen:



Frage: Wieso wurde mod zu mod* modifiziert?

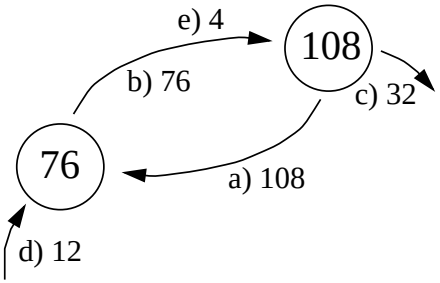
Greife zwei beliebige Elemente a_i, a_j ($i \neq j$) heraus und ersetze den grösseren Wert durch $mod^*(a_i,a_j)$

==> konvergiert gegen den ggT

- Dies nun "verteilt" realisieren! ← In "Teamarbeit", damit es schneller geht...

Nachrichtenaustausch

Prinzip: Kooperation durch Kommunikation



zwischen den Nachbarn auf dem Ring

- Nachrichten sind beliebig lange unterwegs
- Parallele Aktivitäten

Idee:

- Initial beide Nachbarn spontan informieren
- Danach nur auf eintreffende Nachrichten *reagieren*
- Neue Erkenntnisse an alle Nachbarn übermitteln

Verhaltensbeschreibung eines Prozesses P_i:

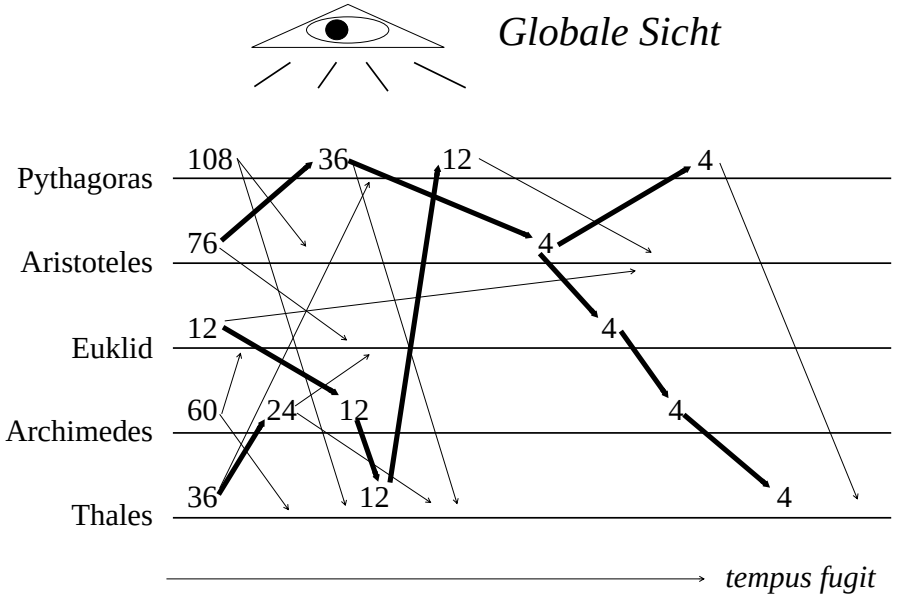
```
{Eine Nachricht <y> ist eingetroffen}
if y < Mi then
   Mi := mod(Mi-1,y)+1;
   send <Mi> to all neighbours;
fi
```

Jeder Prozess P_i hat seine eigene Variable M_i.

Jeder Prozess hat das gleiche prinzipielle Verhalten.

Atomare Aktion:
In der Zwischenzeit u.U. eintreffende Nachrichten werden im "Hausbriefkasten" zwischengepuffert...

Berechnungsablauf und Zeitdiagramm



Ablauf einer *möglichen* "verteilten Berechnung" mit einem *Zeitdiagramm*.

Nicht-deterministisch, abhängig von der Nachrichtenlaufzeit!

- *Terminiert* wenn (?):

  (a) jeder Prozess den ggT kennt,

  (b) alle den gleichen Wert haben,

  (c) alle Prozesse passiv sind und keine Nachricht unterwegs ist.

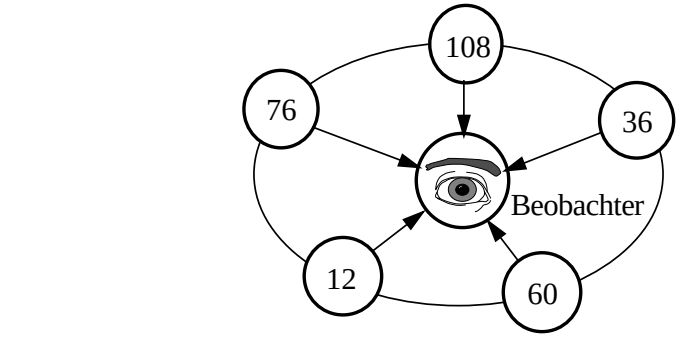
  - Was ist adäquat?
  - Wie feststellen?

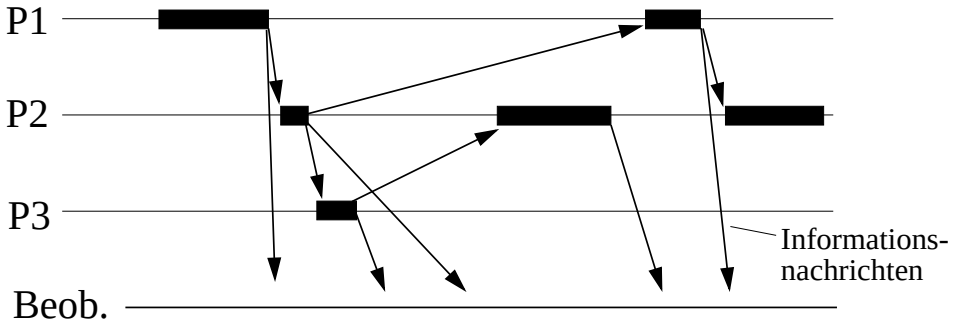
Das ist unrealistisch!

Beweisbare math. Eigenschaft

Diese "stabile" Stagnationseigenschaft ist *problemunabhängig*!

Globaler ggT-Beobachter?





Bekommt der Beobachter ein "richtiges Bild" des Geschehens?

- was heisst das genau?
- und wenn Informationsnachrichten verschieden schnell sind?
- könnte der Beobachter einen Zwischenzustand (z.B. "alle haben den Wert 12") irrtümlich als Endzustand interpretieren?
- wie stellt er überhaupt das Ende der Berechnung fest?