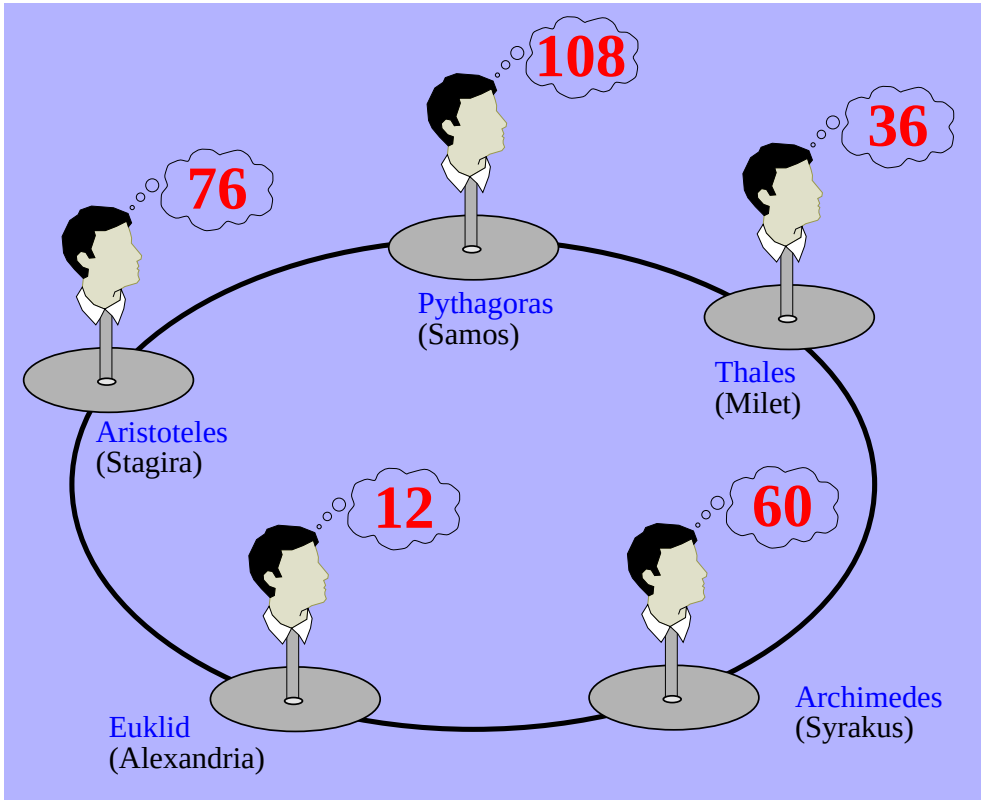
Erste Beispiele für verteilte Algorithmen

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

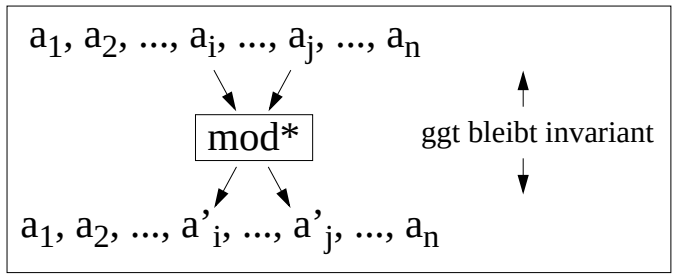
- *Invariante* (für $x \geq y > 0$):

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

  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?

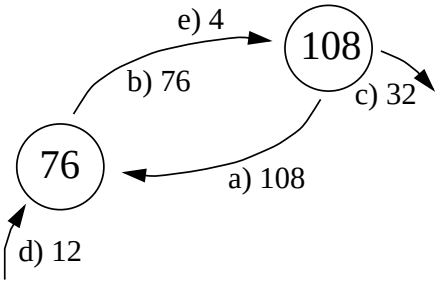
Greife zwei beliebige Elemente a_i, a_j ($i \neq j$) heraus und ersetze den grösseren Wert durch $mod^*(a_i,a_j)$

==> 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:

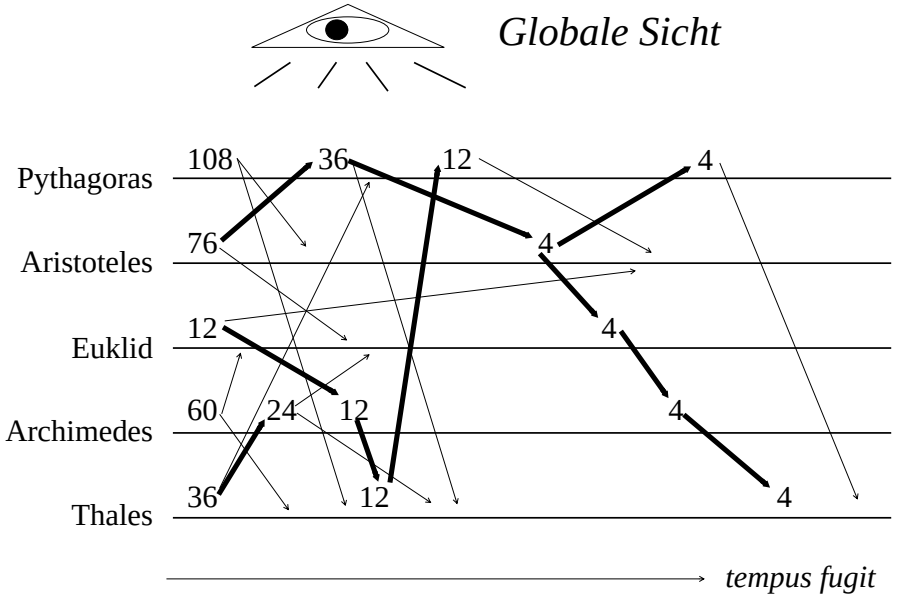
{Eine Nachricht <y> ist eingetroffen}
if $y < M_i$ **then**
  $M_i := mod(M_i\text{-}1,y)+1;$
  send <M_i> **to** all neighbours;
fi

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.

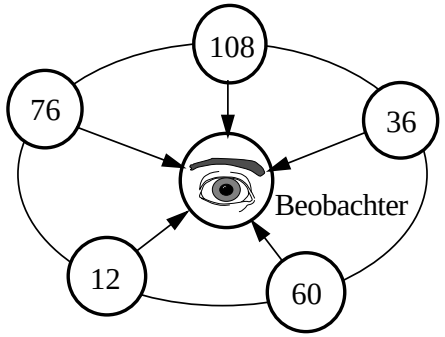
  - Was ist adäquat?
  - Wie feststellen?

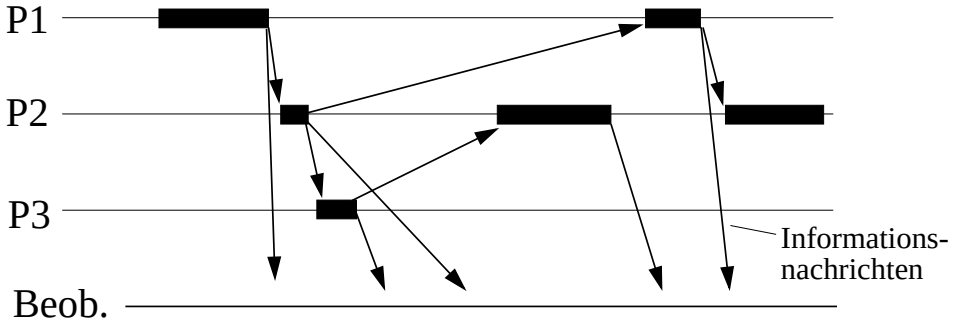
Das ist unrealistisch!

Beweisbare math. Eigenschaft

Diese "stabile" Stagnationseigenschaft ist *problemunabhängig*!

Globaler ggT-Beobachter?





Bekommt der Beobachter ein "richtiges Bild" des Geschehens?

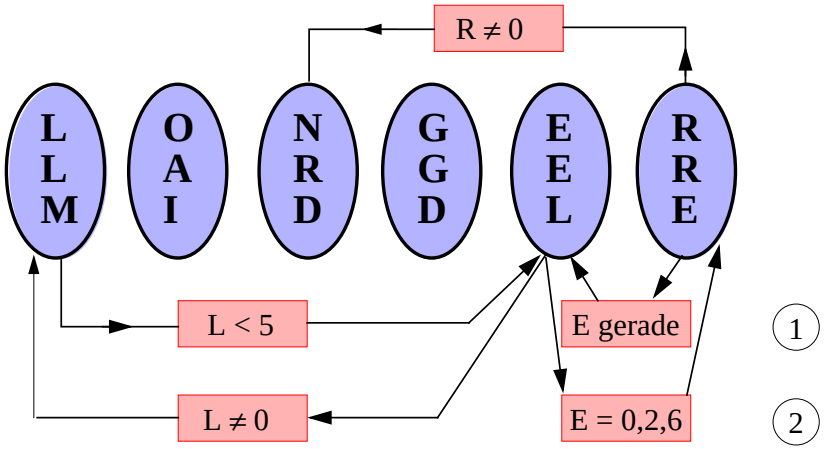
- was heisst das genau?
- und wenn Informationsnachrichten verschieden schnell sind?
- könnte der Beobachter einen Zwischenzustand (z.B. "alle haben den Wert 12") irrtümlich als Endzustand interpretieren?
- wie stellt er überhaupt das Ende der Berechnung fest?

Ein weiteres Beispielproblem: Paralleles Lösen von "Zahlenrätseln"

```
 LONGER          207563
+LARGER    ->   +283563
-------         -------
 MIDDLE          491126
```

Pro Spalte ein Prozess

- Reaktives Verhalten: Auslösen atomarer Aktionen
- Propagieren neuer Erkenntnisse (= Einschränkungen)
  ("parallele Constraint-Propagation")



Endergebnis:

$1 \le L \le 4$
$M \ge 2$
R = 1,3,5,6,8
E = 0,2,6
sonst keine Einschränkung

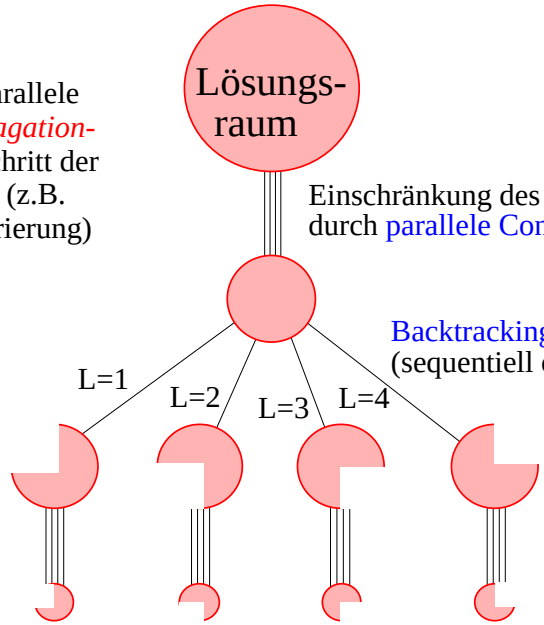
Probleme:

1) Keine eindeutige Lösung
   --> Backtrack-Algorithmus
2) Entdeckung der "Stagnation"?
   (Ende der Parallelphase)

Offensichtlich *gleiches Schema* wie ggT-Berechnung!

Der aufgesetzte Backtrack-Algorithmus

Abwechselnd: Parallele *Constraint-Propagation-Phase* und ein Schritt der *Backtrack-Phase* (z.B. Hypothesengenerierung)



- Hypothese = beliebige Menge von Constraints
  (von einem "Orakel" statt von einer Spalte)

- Einheit, die die Hypothesen generiert und verwaltet, muss die Terminierung einer Constraint-Phase feststellen können
  - wie würde man hier die Terminierung zweckmässigerweise definieren?
  - problembezogen wie bei ggT ("alle Werte identisch") hier nicht einfach
  - "alle passiv und keine sinnvolle Nachricht mehr unterwegs"?

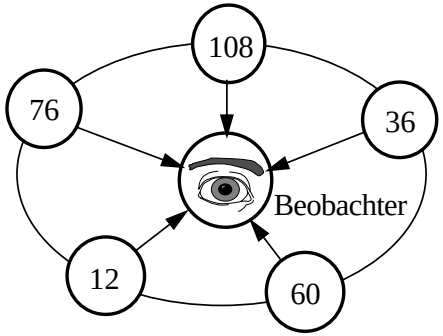
Bemerkungen:

- nicht unbedingt beste Parallelisierungsstrategie!
- Problem ist (in Verallgemeinerung) NP-vollständig

Übungen (1) zur Vorlesung "Verteilte Algorithmen"...

a) Man zeichne Raum-Zeit-Diagramme für verschiedene Abläufe des verteilten ggT-Algorithmus

b) Wie kann man beweisen, dass für *jeden* denkbaren Ablauf das Endergebnis stets der ggT ist?



c) Bleibt der Algorithmus (und/oder der Beweis) korrekt, wenn im Algorithmus $y<M_i$ durch $y \leq M_i$ ersetzt wird?

Verhaltensbeschreibung eines Prozesses P_i:

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

...Übungen (1)

d) Man vergleiche die verteilte Berechnung des ggT-Algorithmus für zwei Zahlen mit dem üblichen sequentiellen ggT-Algorithmus für zwei Zahlen

e) Genügt es auch, nur in Uhrzeigerrichtung eine Nachricht zu senden (anstatt an beide Nachbarn)?

f) Kann statt des Ringes eine andere Topologie verwendet werden? Welche?

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

h) Wie kann man erreichen, dass ein ggT-Beobachter (der über jede Wertänderung eines Prozesse informiert wird) eine "kausaltreue" Beobachtung macht?

i) Beobachtungen sind eine lineare Ordnung von (beobachteten) Ereignissen. In welcher Beziehung steht die oben erwähnte Halbordnung zu dieser linearen Ordnung? Können Sie eine Vermutung darüber anstellen, was der Schnitt aller möglichen kausaltreuen Beobachtungen einer verteilten Berechnung aussagt?

j) Wie kann der Beobachter die Terminierung erkennen?