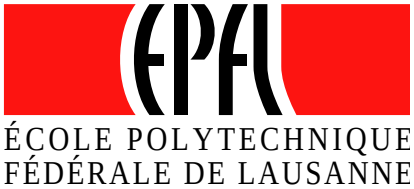
Entdecken globaler Prädikate in fehlerbehafteten verteilten Systemen

Felix Gärtner



École Polytechnique Fédérale de Lausanne (EPFL)
Departement de Systémes de Communications
Laboratoire de Programmation Distribuée
CH-1015 Lausanne, Schweiz

fgaertner@lpdmail.epfl.ch

Motivation: Abhängigkeit von Computersystemen

Quelle: http://www.cs.virginia.edu/~survive/

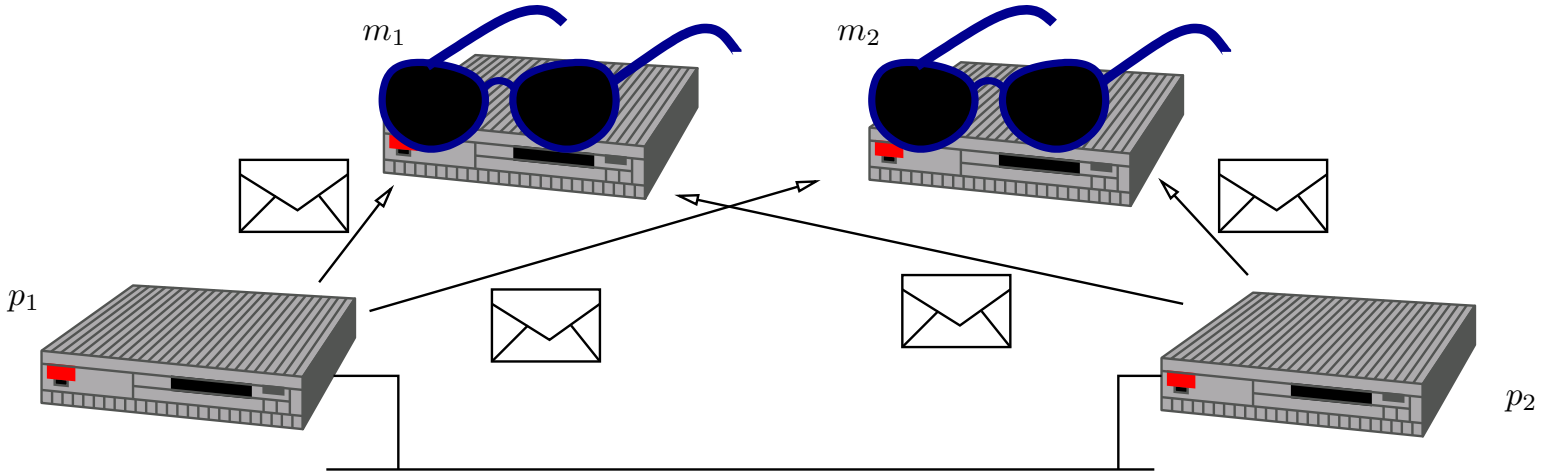
Warum Prädikatserkennung?

- Entdeckung von Prädikaten auf dem Systemzustand ist der Kern vieler verteilter Kontrollaufgaben:
  - Systemkontrolle in Kraftwerken: Ist der Turbinendruck kritisch und noch kein Ventil geöffnet?
  - Verteiltes Debugging: Galt jemals $counter < maxsize$?
  - Algorithmenkomposition: Ist erster Algorithmus terminiert?
- Wie kann man verteilte Systeme sinnvoll beobachten? (Stand der Forschung)
- Was ändert sich, wenn Fehler auftreten können?
- Abstrakte Sichtweise, zielt auf prinzipielle Zusammenhänge.

Probleme beim Beobachten

Wie beobachtet man verteilte Systeme?

- Kommunikation über Nachrichten.
- Beobachtetes System: Rechner und Kommunikationsnetzwerk.
- Beobachtendes System: Überwachungsrechner ("Monitore"), die möglichst über ein dezidiertes Netzwerk an das beobachtete System angeschlossen sind.



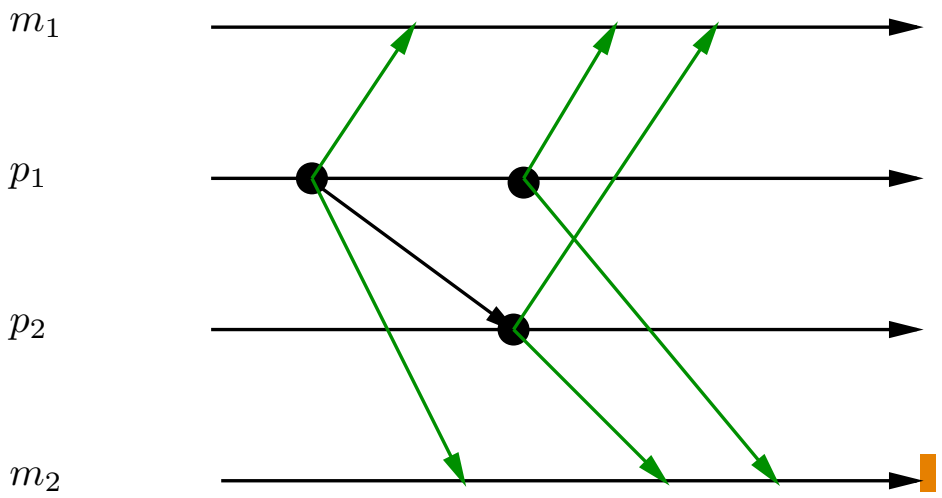
- Welche Annahmen werden bezüglich des Zeitverhaltens gemacht?

Das asynchrone Systemmodell

- Ziel: möglichst einfaches und doch realistisches Modell.
- Ansatz: möglichst wenig Annahmen über das Zeitverhalten.
- Zeitverhalten von Nachrichten:
  - Nachrichten können lange unterwegs sein.
- Zeitverhalten von Rechnern:
  - Computer können sehr langsam werden.
- Gebräuchliches Modell für das Internet.
- Zunächst: keine Fehler im System (keiner stürzt ab, keine Nachrichten gehen verloren, usw.).
- Was heißt jetzt "Beobachten" im asynchronen Modell?

Raum/Zeit-Diagramme und Beobachtungen

- Jeder Prozeß hat eigene Zeitachse (Zeit vergeht nach links). Lokale Ereignisse und Nachrichten als Punkte und Pfeile einzeichnen.

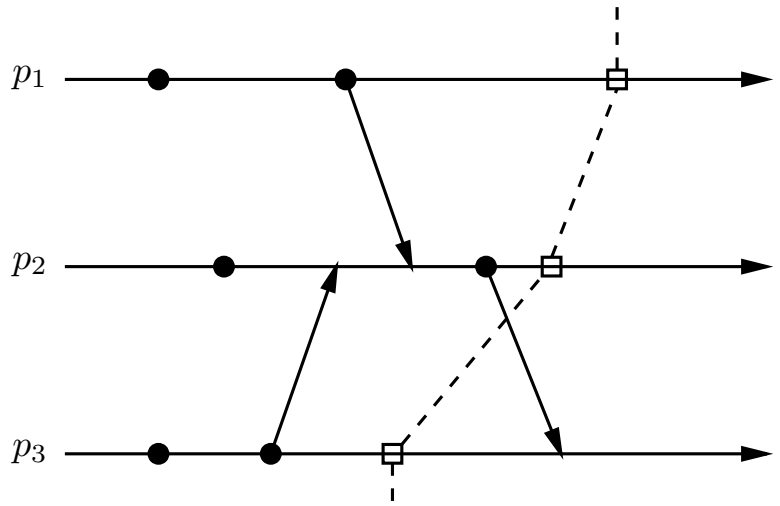


- Monitorprozesse konstruieren aus den einzelnen (grünen) Kontrollnachrichten eine Beobachtung (Sequenz von globalen Zuständen).

- Beobachten: Erkennen, ob in der Beobachtung ein Prädikat auf dem globalen Zustand gilt . Aber: Was bedeutet "gilt"? Was ist ein "globaler Zustand"?

Globaler Zustand

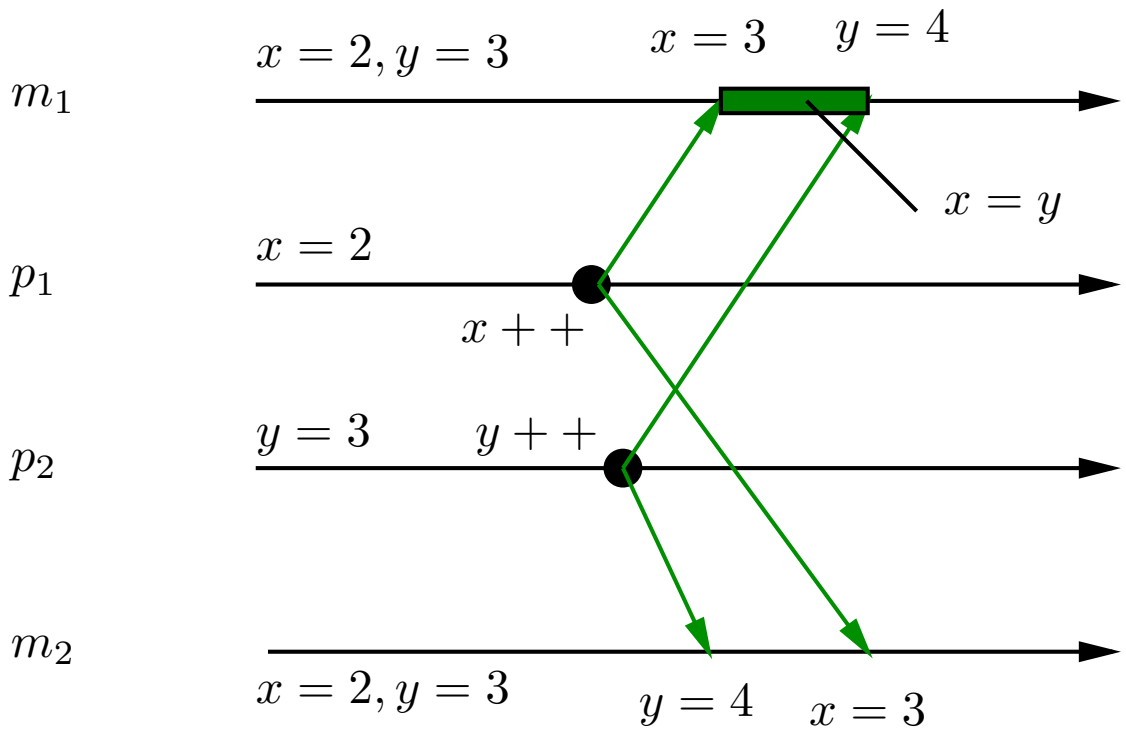
- Globaler Zustand: Vektor von allen lokalen Prozeßzuständen (und Nachrichten, die unterwegs sind)
  - Kann Nachrichten als Teil des Prozeßzustandes auffassen (verschickt, aber noch nicht empfangen)
  - lokale Zeitpunkte, an denen Zustand gespeichert wird, definieren einen Schnitt (*cut*) durch die Berechnung



- Verwandtes Problem: Schnappschußproblem.

Unterschiedliche Beobachtungen

- Definition globaler Zustand geklärt. Bei Definition von "gilt" Vorsicht vor Beobachterabhängigkeit!



- Die Gültigkeit eines Prädikats ist abhängig von der Beobachtung, nicht von der Berechnung. Wie kann man dieses Phänomen charakterisieren?

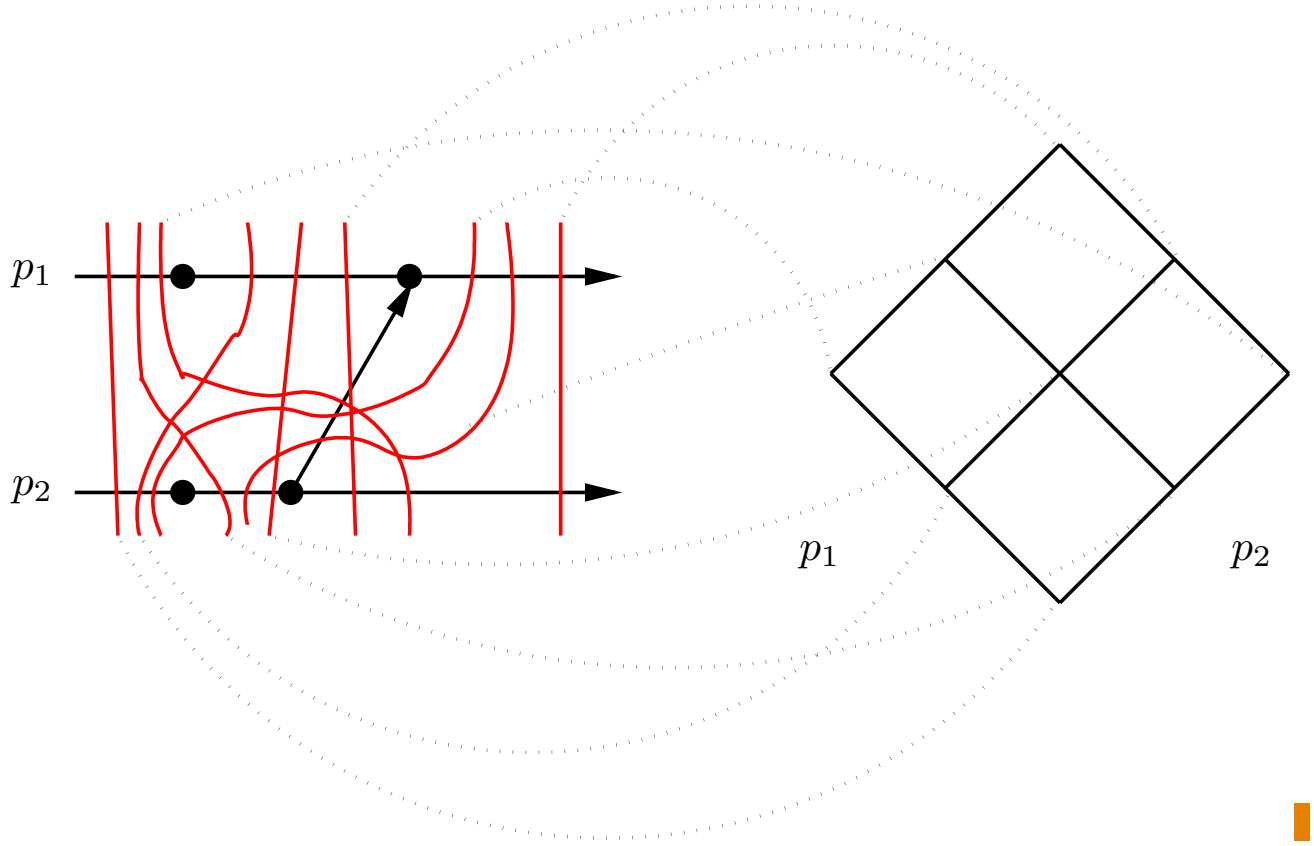
Verband der globalen Zustände [Mattern 1989]

- Betrachte die Menge M aller globalen Zustände der Berechnung.
- Betrachte zudem die (transitive) Relation $\leq$ ("geht hervor aus") über globalen Zuständen.

$\Rightarrow$ $(M, \leq)$ bildet einen Verband.

- Beobachtung ist ein Pfad durch diesen Verband (Verband repräsentiert alle möglichen Beobachtungen).

Beispiel



Formale Definition: Prädikatsentdeckung

- Gegeben ein Prädikat φ auf globalen Zuständen einer verteilten Berechnung C und eine Definition von "φ gilt in C".

- Ein verteilter Algorithmus für Prädikatsentdeckung hat folgende Eigenschaften:
  - *liveness*: Wenn φ in der Berechnung gilt, dann wird der Algorithmus dies auch nach endlicher Zeit melden.
  - *safety*: Falls der Algorithmus die Gültigkeit von φ meldet, dann gilt φ auch in der Berechnung.

- Verschiedene Instanzen und Lösbarkeiten des Problems möglich.

Modalitäten und stabile Prädikate

- "φ gilt" kann man beobachterunabhängig auf dem Verband der globalen Zustände definieren:
  - $possibly(\varphi)$ gilt falls es eine Beobachtung gibt, in der φ wahr ist.
  - Anwendung: Fehlerbedingung im *debugger*.
  - $definitely(\varphi)$ gilt falls in jeder Beobachtung φ gilt.
  - Anwendung: Terminierungsbedingung einer Transaktion.
- Alternative: Art der Prädikate einschränken.
  - Stabile Prädikate: einmal wahr, immer wahr.
  - Entdecken stabiler Prädikate ist beobachterunabhängig.
  - Bekannter Algorithmus von Chandy and Lamport [1985].
  - Beobachterunabhängige Prädikate [Charron-Bost et al. 1995].

Ausblick

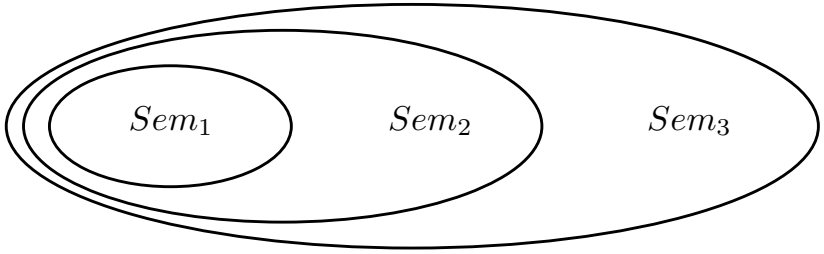
- Fazit: Beobachten asynchroner verteilter Systeme ist schwierig/komplex.
- Aber es gibt mittlerweile gut erforschte Konzepte für den fehlerfreien Fall.
- Jetzt: Wie kann man die Konzepte und Verfahren übertragen in Systeme, in denen Fehler auftreten können.
- Fehlerannahme = genaue Beschreibung, welche Komponenten wie ausfallen können.
- *crash* Fehlerannahme = höchstens t Prozesse stürzen ab (hören auf, ihren lokalen Algorithmus auszuführen).

Semantiken der Prädikatsentdeckung

- Perfekte Prädikatsentdeckung Sem_1 (wie oben):
  - (S) Falls der Algorithmus die Gültigkeit von φ meldet, dann galt es auch in der Berechnung.
  - (L) Falls φ gilt, wird der Algorithmus dies auch nach endlicher Zeit melden.

- Stabilisierende Prädikatsentdeckung Sem_2: L and $\Diamond S$.
  - Nur endlich viele falsche Verdächtigungen.

- *best effort* Prädikatsentdeckung Sem_3:
  - L and $\Box\Diamond S$.
  - Man wird ein Prädikat, welches nie galt, nicht dauerhaft melden.



Neue Arten von Prädikaten

- Fehler sollten auch in Prädikaten ausdrückbar sein: Prädikat up_i bezieht sich auf den operationellen Zustand von p_i.

- Kann up_i in Prädikaten verwenden:
  - Prozeß p_i stürzte nach dem vierten Ereignis ab:

$$\neg up_i \wedge ec_i = 4$$

  - Jeder Prozeß hat ein *commit* ausgeführt oder stürzte ab:

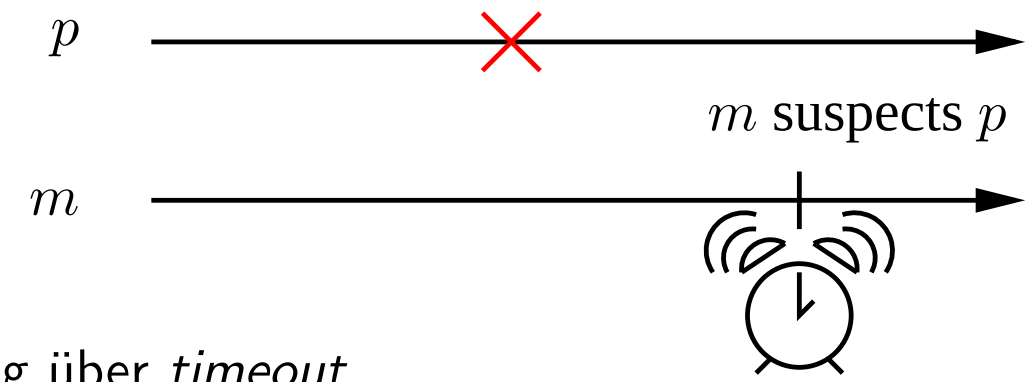
$$\forall i : \neg up_i \vee commit_i = 1$$

  - Prozeß p_i wartet auf eine Nachricht von einem abgestürzten Prozeß:

$$j \in waiting_i \wedge \neg up_j$$

- Wie kann man up_i aktuell halten?

Fehlerdetekoren



- Implementierung über *timeout*.
- Fehlerdetektor "verdächtigt" einen anderen Prozeß, abgestürzt zu sein.
- Gewünsche Eigenschaft (ohne Bezug auf Implementierung):
  - Prozeß p wird nicht verdächtigt, bevor er abgestürzt ist.
  - Falls p abstürzt, wird er nach endlicher Zeit dauerhaft verdächtigt.
- Klasse der perfekten Fehlerdetectoren $\mathcal{P}$.

Stabilisierende Versionen von $\mathcal{P}$

- Perfekte Fehlerdetektoren kann man nicht in asynchronen Systemen implementieren [Fischer et al. 1985; Chandy and Misra 1986].
- Setzen eines guten *timeout* Wertes ist schwierig (in der Praxis sind Fehlerdetekoren meistens nicht perfekt).
- Nach endlicher Zeit perfekt $\Diamond\mathcal{P}$ [Chandra and Toueg 1996]:
  - Es gibt eine Zeit nach der kein Prozeß fälschlicherweise verdächtigt wird.
- Unendlich oft genau $\Box\Diamond\mathcal{P}$ [Garg and Mitchell 1998]:
  - Prozesse, die nicht abstürzen, werden nicht dauerhaft verdächtigt.

Endeckungsalgorithmus

boolean variable $history$ initially false

upon $\langle$control message arrives or failure detector information changes$\rangle$
do
  $\langle$update own perception of global state$\rangle$
  if $\langle\varphi$ holds on global state$\rangle \wedge \neg history$ **then**
    $history := true$
    $\langle$trigger detection event$\rangle$
  elsif $\langle\neg\varphi$ holds on global state$\rangle \wedge history$ **then**
    $history := false$
    $\langle$trigger undetection event$\rangle$
  end

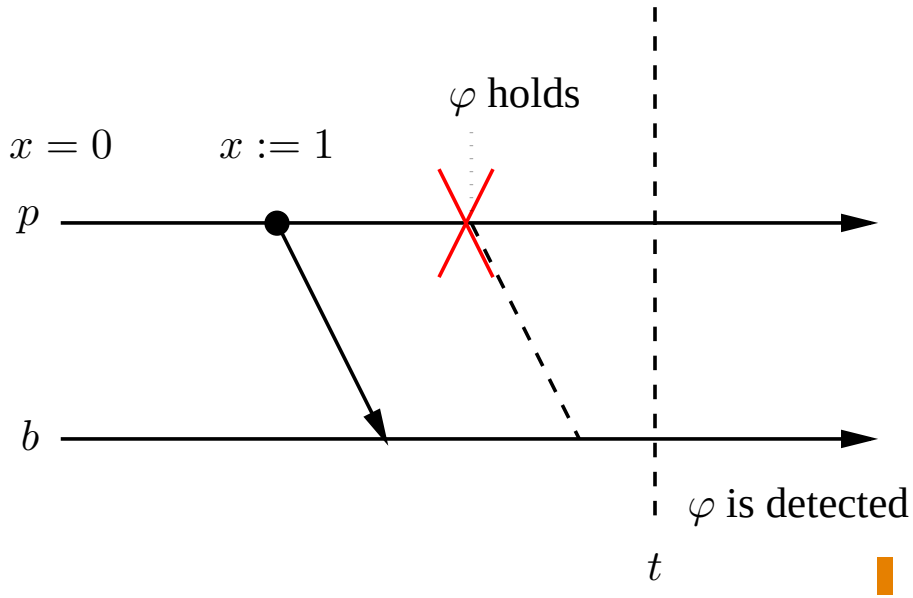
- Randbemerkung: Kontrollnachrichten müssen in kausaler Ordnung zugestellt werden.

Prädikatsentdeckung: Möglichkeiten [Gärtner and Pleisch 2001]

- Stabilisierende Prädikatsentdeckung Sem_2 möglich bei Verwendung von $\Diamond\mathcal{P}$.
  - Stabilisierung des Fehlerdetektors genügt, um nur endlich viele falsche Entdeckungen zu erhalten.
- *best effort* Prädikatsentdeckung Sem_3 erreichbar bei Verwendung von $\Box\Diamond\mathcal{P}$.
  - Unendlich häufige Genauigkeit von $\Box\Diamond\mathcal{P}$ vermeidet permanente falsche Verdächtigung eines nicht abgestürzten Prozesses.
- Frage: Kann man mit $\mathcal{P}$ perfekte Prädikatsentdeckung implementieren?

Unmöglichkeit perfekter Prädikatsentdeckung mit Fehlerdetektoren [Gärtner and Pleisch 2002]

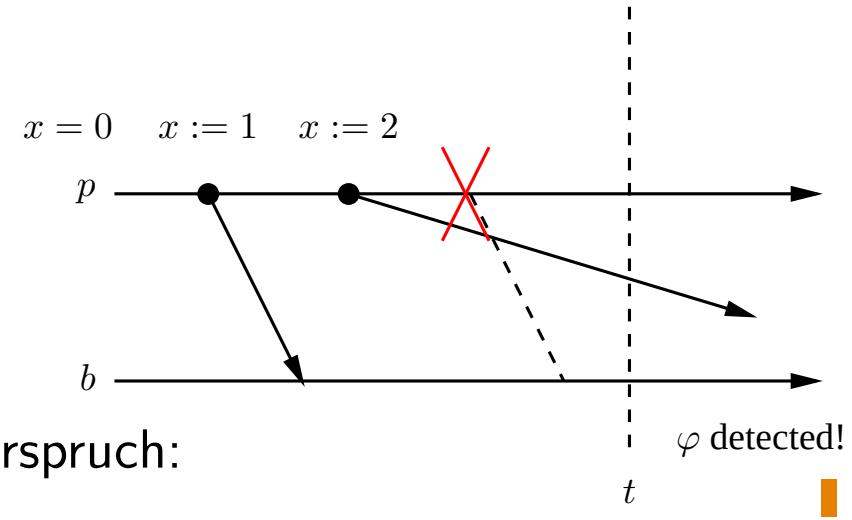
- Angenommen es gibt einen Algorithmus A, der mit Hilfe eines Fehlerdetektors perfekte Prädikatsentdeckung löst.
  - Betrachte das Prädikat $\varphi \equiv crashed \wedge x = 1$ in folgendem Szenario:



- Da A perfekte Prädikatsentdeckung löst und φ gilt, wird A nach endlicher Zeit (Zeitpunkt t) die Gültigkeit melden.

Beweis (Fortsetzung)

- Betrachte ähnliches Szenario:



- Beweis durch Widerspruch:
  - In diesem Szenarion passiert ein Ereignis $x := 2$ nach $x := 1$ und dem Absturz.
  - Das System ist asynchron $\Rightarrow$ Kontrollnachricht kann sich beliebig lange verzögern.
  - Für Prozeß b sieht zur Zeit t dieses Szenario genaus aus wie das erste Szenario (ein Fehlerdetektor hilft hier nicht bei der Unterscheidung).
  - A ist deterministisch: A meldet die Entdeckung von φ zur Zeit t. Aber: φ hat nie gegolten, A kann nicht korrekt sein.

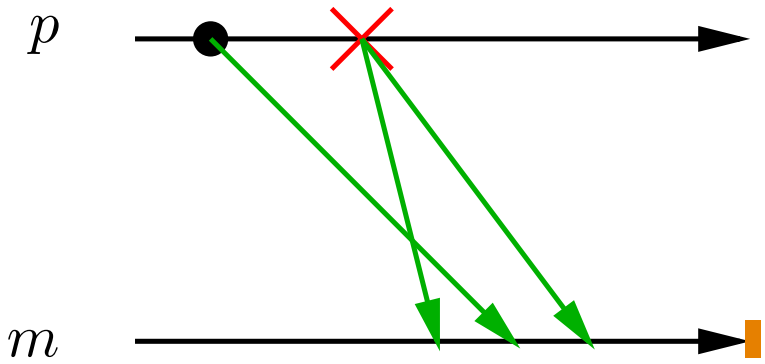
Wege aus der Unmöglichkeit

- Mit Fehlerdetektoren kann man bestenfalls Sem_2 erreichen (für allgemeine Prädikate).

- Drei Alternativen, um perfekte Prädikatsentdeckung zu ermöglichen:
  - "Gültigkeit" anders definieren: Modalitäten, die Ungenauigkeit der Fehlerdetektoren miteinbeziehen [Gärtner and Kloppenburg 2000].
  - Einschränkung auf bestimmte Prädikatsklassen: Wenn φ stabil ist, reicht $\mathcal{P}$ aus für perfekte Prädikatsentdeckung [Gärtner and Pleisch 2001].
  - Erweiterung des Fehlerdetektorkonzeptes: *failure detection sequencer* [Gärtner and Pleisch 2002] (erst dann verdächtigen, wenn alle Nachrichten angekommen sind).

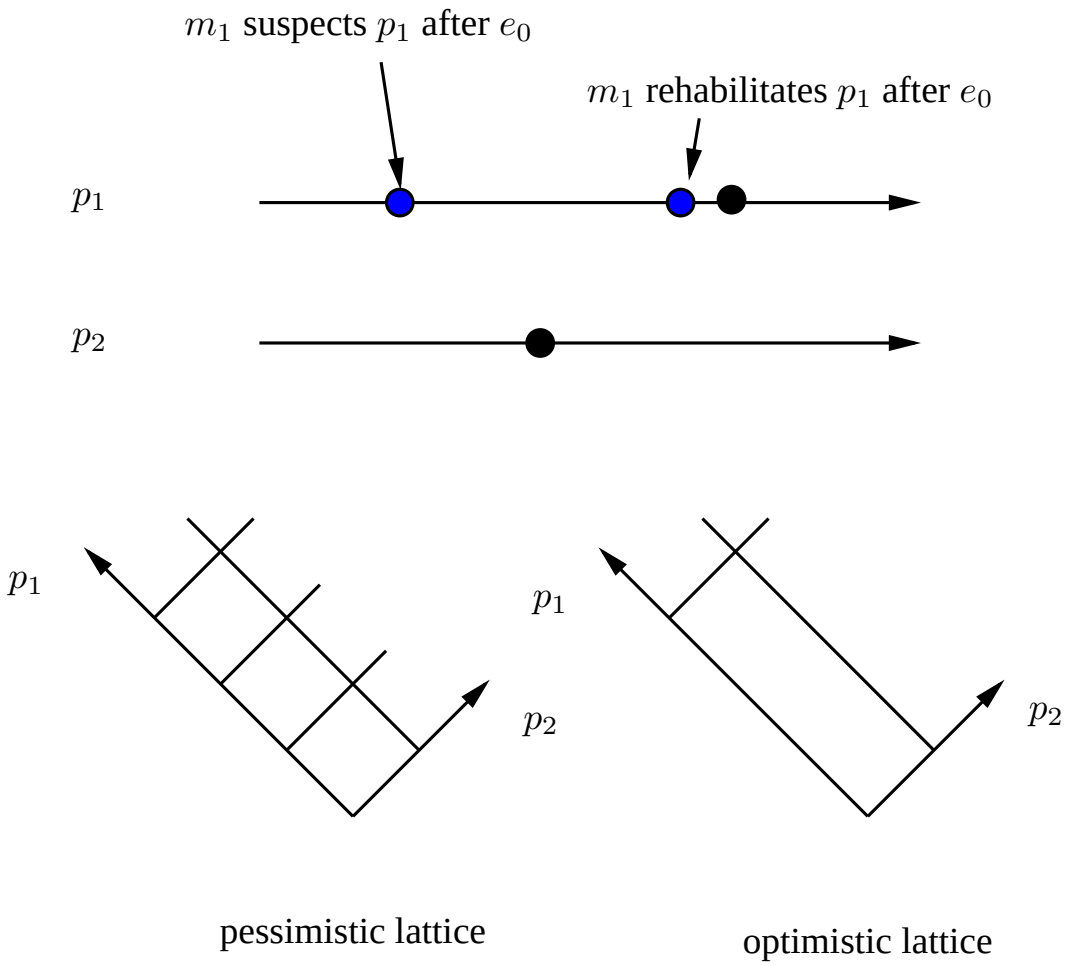
Idee des *sequencers*

- Perfekte Prädikatsentdeckung mit einem bestimmten *sequencer* Σ:
  - Standardtechniken zur Prädikatsentdeckung in fehlerfreien Systemen benutzen.
  - Mit Σ die *crash* Ereignisse in die beobachtbare Kausalordnung einsortieren.
- Man kann Ereignis in die korrekte "Sequenz" bringen.



- Erkennen der kausalen Ordnung der Fehlerereignisse.

Idee abgeschwächter Modalitäten



pessimistic lattice

optimistic lattice

Zusammenfassung

- Das Entdecken globaler Prädikate ist ein grundlegendes und komplexes Problem in verteilten Systemen.
  - Grundlegende Einsichten: Verband globaler Zustände (und Konsistenzdefinition basierend auf Kausalität).
- Problem wird schwieriger, wenn Fehler auftreten können.
  - Bisher nur *crash* Fehler untersucht und die Rolle von Fehlerdetektoren umrissen.
- Untersuchungen geben fundamentale Einsichten in die möglichen Effekte von Fehlern auf die Schwierigkeit von Problemen.
- Kern der Methodik: Finde sinnvolle (auch dem Praktiker erläuterbare) Abstraktionen (wie z.B. Fehlerdetektoren).
- Untersuchungen stehen erst am Anfang (Raum möglicher Fehlerannahmen ist noch sehr groß).

Acknowledgments

- Slides produced using pdfLaTeX and Klaus Guntermann's PPower4.

References

Chandra, T. D. and Toueg, S. 1996. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM 43*, 2 (March), 225–267.

Chandy, K. M. and Lamport, L. 1985. Distributed snapshots: determining global states of distributed systems. *ACM Transactions on Computing Systems 3*, 1, 63–75.

Chandy, K. M. and Misra, J. 1986. How processes learn. *Distributed Computing 1*, 40–52.

Charron-Bost, B., Delporte-Gallet, C., and Fauconnier, H. 1995. Local and temporal predicates in distributed systems. *ACM Transactions on Programming Languages and Systems 17*, 1 (Jan.), 157–179.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM 32*, 2 (April), 374–382.

Garg, V. K. and Mitchell, J. R. 1998. Distributed predicate detection in a faulty environment. In *Proceedings of the 18th IEEE International Conference on Distributed Computing Systems (ICDCS98)* (1998).

Gärtner, F. C. and Kloppenburg, S. 2000. Consistent detection of global predicates under a weak fault assumption. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS2000)* (Nürnberg, Germany, Oct. 2000), pp. 94–103. IEEE Computer Society Press.

Gärtner, F. C. and Pleisch, S. 2001. (Im)Possibilities of predicate detection in crash-affected systems. In *Proceedings of the 5th Workshop on Self-Stabilizing Systems (WSS2001)*, Number 2194 in Lecture Notes in Computer Science (Lisbon, Portugal, Oct. 2001), pp. 98–113. Springer-Verlag.

Gärtner, F. C. and Pleisch, S. 2002. Failure detection sequencers: Necessary and sufficient information about failures to solve predicate detection. In *Proceedings of the 16th International Symposium on DIStributed Computing (DISC 2002)* (Toulouse, France, Oct. 2002).

Mattern, F. 1989. Virtual time and global states of distributed systems. In M. C. et al. Ed., *Proceedings of the International Workshop on Parallel and Distributed Algorithms* (Chateau de Bonas, France, 1989), pp. 215–226. Elsevier Science Publishers. Reprinted on pages 123–133 in [Yang and Marsland 1994].

Yang, Z. and Marsland, T. A. Eds. 1994. *Global States and Time in Distributed Systems*. IEEE Computer Society Press.

Zusatzfolien

Konsistente und inkonsistente Zustände

- Zustand (Schnitt) ist *konsistent* wenn er zu jeder Wirkung auch die Ursache enthält.