Consistent Detection of Global Predicates under a Weak Fault Assumption

Felix Gärtner and Sven Kloppenburg

Darmstadt University of Technology, Germany, felix@informatik.tu-darmstadt.de

Systeam Engineering, Darmstadt, Germany, sven@syseng.de

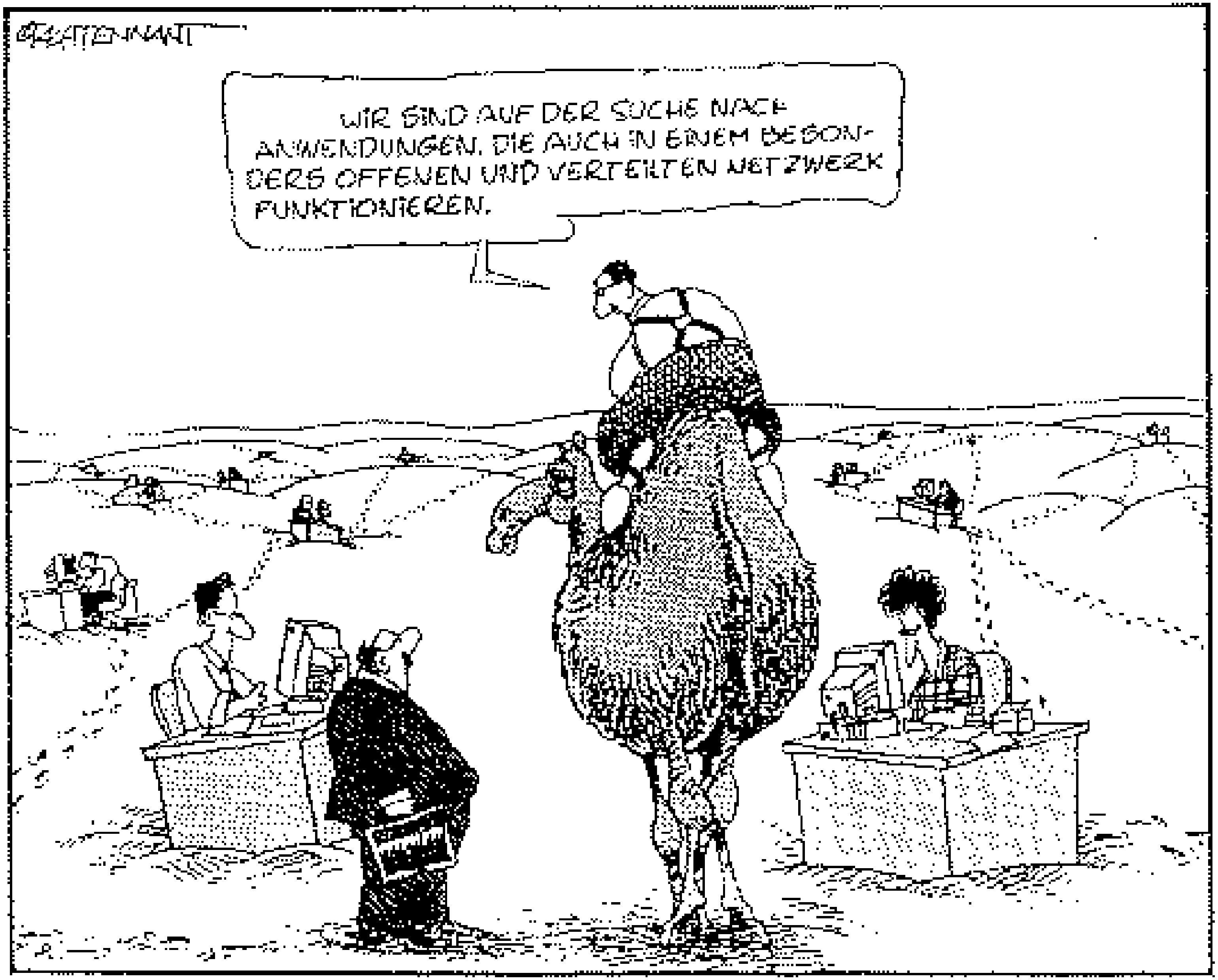
Consistent Detection of Global Predicates under a Weak Fault Assumption

Felix Gärtner and Sven Kloppenburg

Darmstadt University of Technology, Germany, felix@informatik.tu-darmstadt.de

Systeam Engineering, Darmstadt, Germany, sven@syseng.de

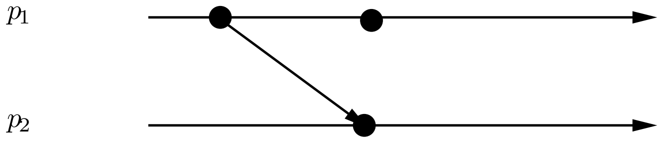
Athene: Godess of wisdom, guardian of arts and crafts (Keynote by Mike Morganti yesterday)


WIR SIND AUF DER SUCHE NACH ANWENDUNGEN, DIE AUCH IN EINEM BESONDERS OFFENEN UND VERTEILTEN NETZWERK FUNKTIONIEREN.

“We are looking for software which also works in *very* large and *very* open distributed systems.”

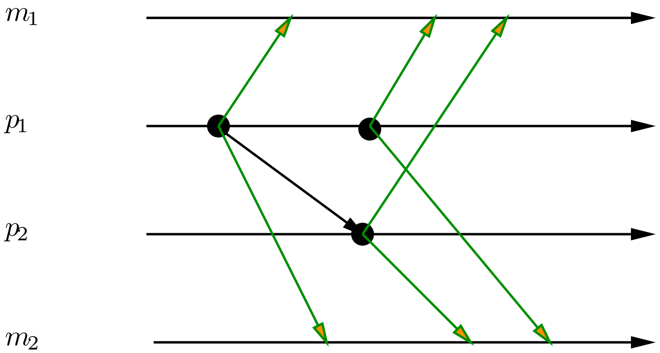
Observation in fault-free asynchronous systems

- Distributed computations in asynchronous systems.



Observation in fault-free asynchronous systems

- Distributed computations in asynchronous systems.



- Application and monitor processes.
- Application and control messages.
- Predicate detection: Lattice of consistent global states.
- Modalities *possibly* and *definitely*.

Predicate detection in faulty asynchronous systems

- crash fault assumption $=$ at most t processes simply stop executing steps.
- For the moment: restrict crash faults to application processes only (monitors always stay alive).
- Predicate up_i refers to functional state of p_i.
- Can be used in predicates:
  - Process p_i crashed after 4th event: $\neg up_i \wedge ec_i = 4$
  - Every process either commits or crashes: $\forall i : \neg up_i \vee commit_i$
- Idea: find suitable analogies to *possibly* and *definitely* for these types of predicates.

Implementable failure detection

- Every monitor must keep up_i up to date (failure detection, discussed in detail by Mikel Larrea yesterday).

- Can ensure eventual detection, but cannot avoid false suspicions.

- Terminology: failure detectors *suspect* and *rehabilitate* application processes.

- Best we can do: a non-crashing process is not permanently suspected [3].

- For observation purposes: add causality information to suspicions:
  - "m_j suspects p_i after event e_k on p_i."
  - "m_j rehabilitates p_i after event e_k on p_i."

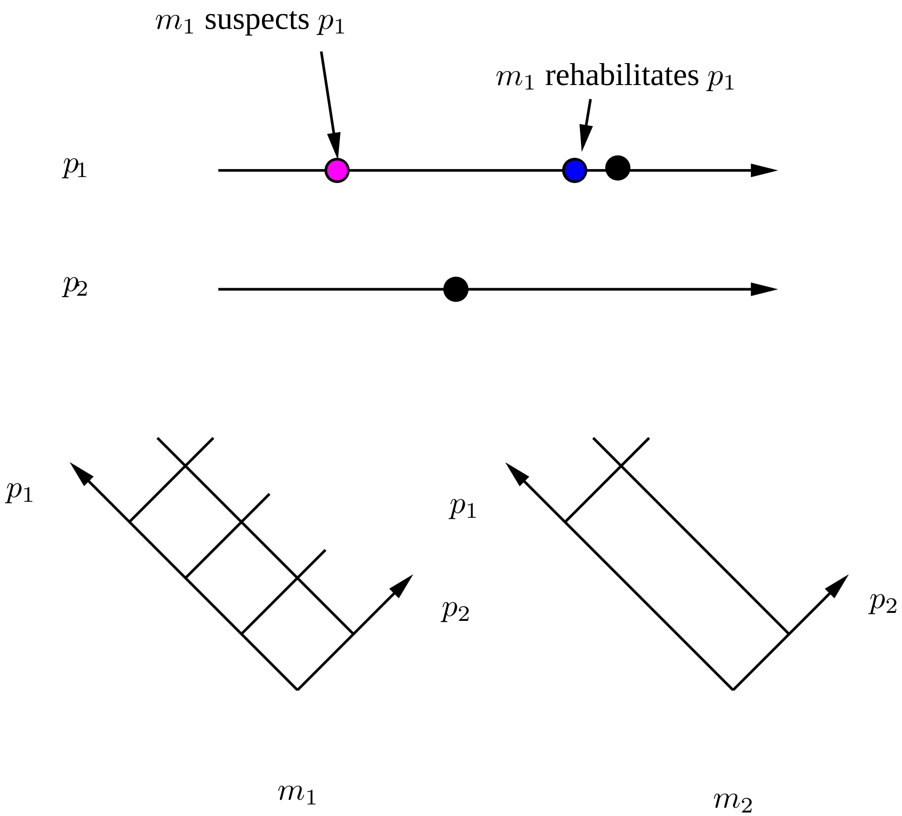
- Assume: between two events at most one suspicion and rehabilitation.

Lattice over extended state space

- Treat up_i as a variable on p_i.
- Suspicion/rehabilitation is a simple state change of p_i (extended state space).
- Change of up in consistent states yields again consistent states.
- Lemma: Integration of suspicions/rehabilitations into state lattice yields new lattice (over extended state space).
- Use this lattice for predicate detection.

Per monitor lattice

- Due to false suspicions monitors construct different state lattices.
- *possibly*/*definitely* not observer-invariant.



Global failure detector semantics

- Problem: false suspicions.

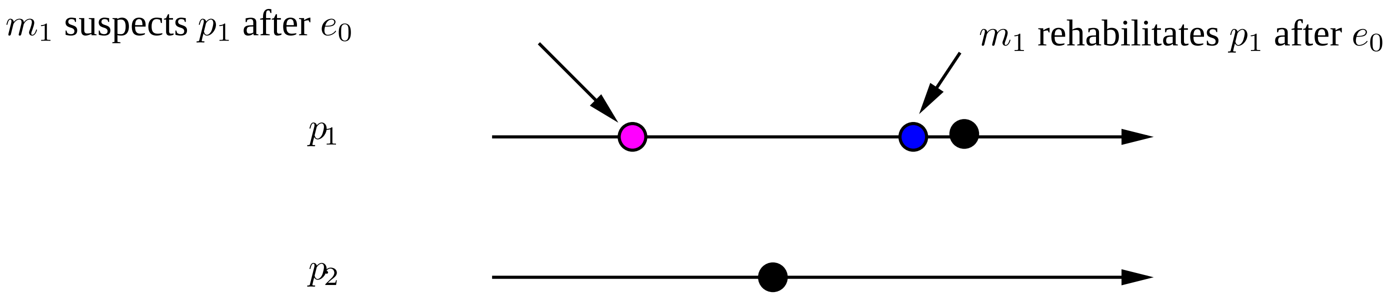
- Solution: define "global" failure detector semantics.

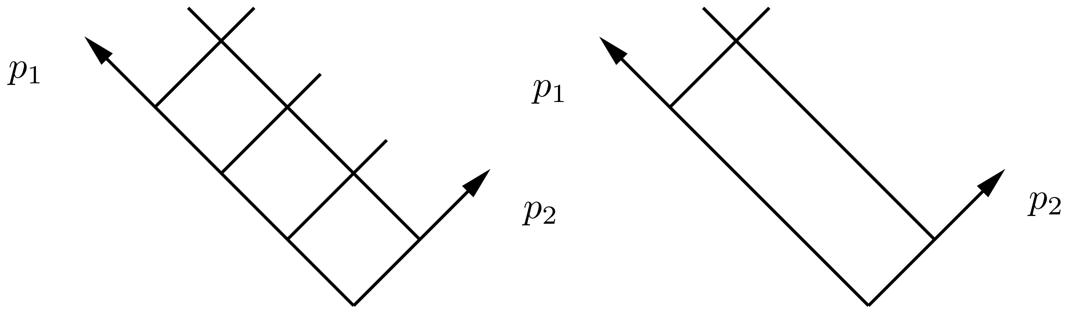
- p_i is (globally) suspected after e_k iff . . .
  - (pessimistic) $\exists$ a monitor which suspects p_i after e_k.
  - (optimistic) $\forall$ monitors suspect p_i after e_k.

- Can define pessimistic and optimistic state lattice (union and intersection of all monitor lattices).

New modalities

- Given predicate φ on extended state space.
- $negotiably(\varphi)$ holds iff $possibly(\varphi)$ holds on pessimistic state lattice.
- $discernibly(\varphi)$ holds iff $definitely(\varphi)$ holds on optimistic state lattice.


m_1 suspects p_1 after e_0
m_1 rehabilitates p_1 after e_0
p_1
p_2

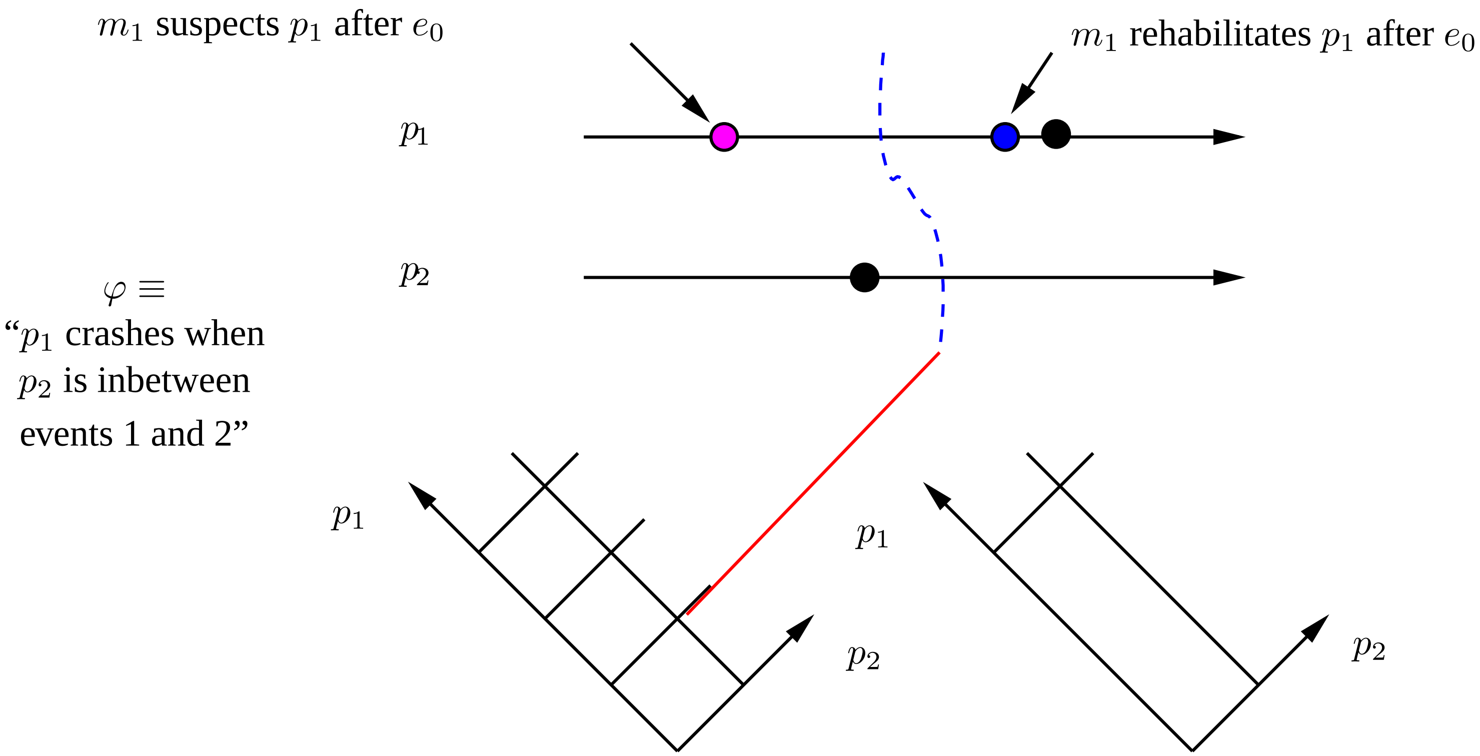


New modalities

- Given predicate φ on extended state space.

- $\mathit{negotiably}(\varphi)$ holds iff $\mathit{possibly}(\varphi)$ holds on pessimistic state lattice.

- $\mathit{discernibly}(\varphi)$ holds iff $\mathit{definitely}(\varphi)$ holds on optimistic state lattice.



m_1 suspects p_1 after e_0

m_1 rehabilitates p_1 after e_0

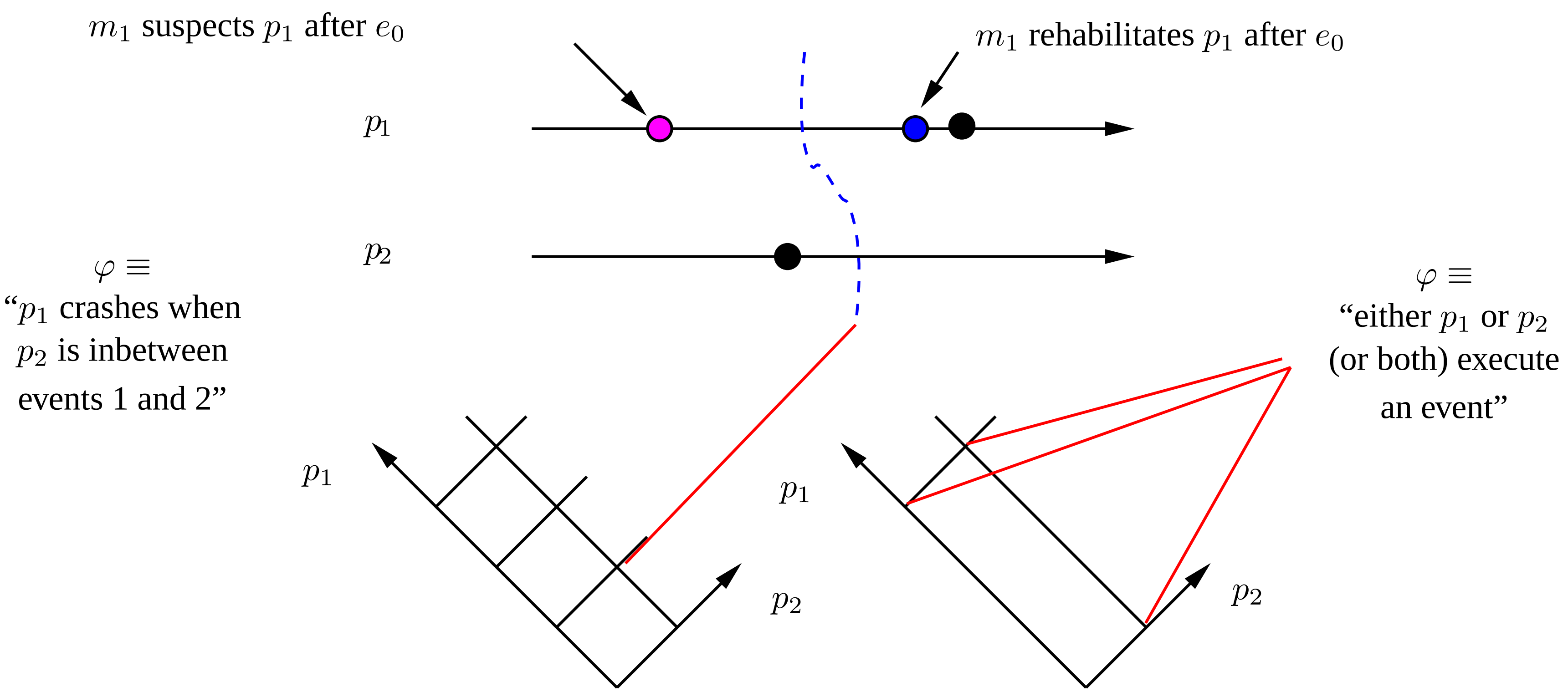
$\varphi \equiv$
"p_1 crashes when
p_2 is inbetween
events 1 and 2"

New modalities

- Given predicate φ on extended state space.

- $\mathit{negotiably}(\varphi)$ holds iff $\mathit{possibly}(\varphi)$ holds on pessimistic state lattice.

- $\mathit{discernibly}(\varphi)$ holds iff $\mathit{definitely}(\varphi)$ holds on optimistic state lattice.



m_1 suspects p_1 after e_0

m_1 rehabilitates p_1 after e_0

$\varphi \equiv$ "p_1 crashes when p_2 is inbetween events 1 and 2"

$\varphi \equiv$ "either p_1 or p_2 (or both) execute an event"

Intuition behind new modalities

- Optimistic/pessimistic lattice can be understood in analogy to optimistic/pessimistic network protocols:
  - pessimistic: be careful all the time, take immediate action if something bad has possibly happened.

  ⇒ use *negotiably* to trigger action.
  - optimistic: go ahead without synchronization and hope for the best, deal with conflicts only when necessary.

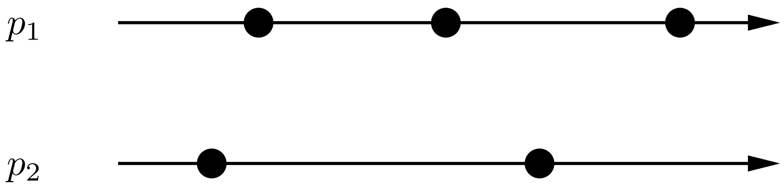
  ⇒ use *discernibly* to ignore spurious suspicions.

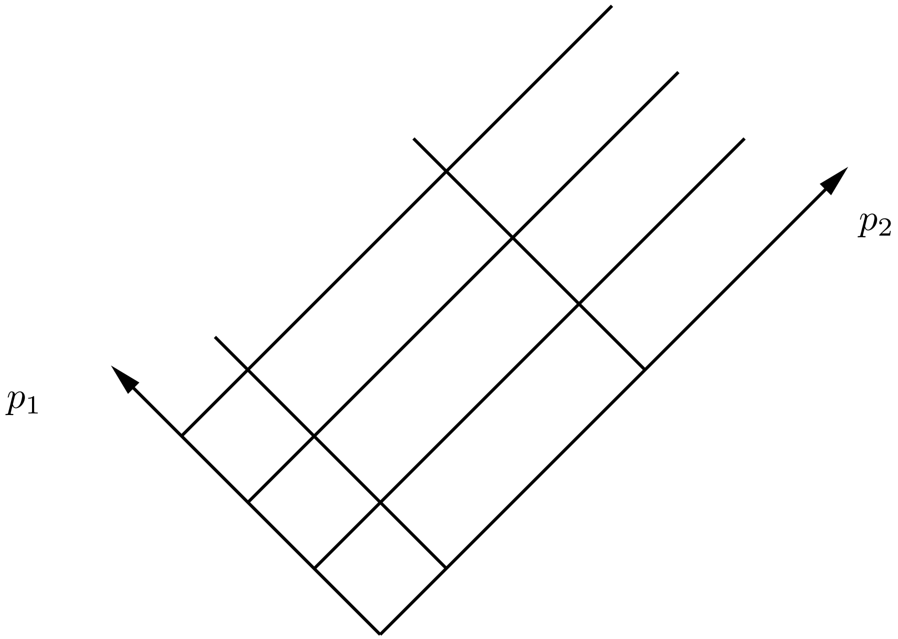
- Understandable in analogy to *possibly*/*definitely*:
  - Safety requirement $\Box\varphi$: take action if $\mathit{negotiably}(\neg\varphi)$ is detected.
  - Liveness requirement $\Diamond\varphi$: validated if $\mathit{discernibly}(\varphi)$ is detected.

Detection algorithms in a nutshell

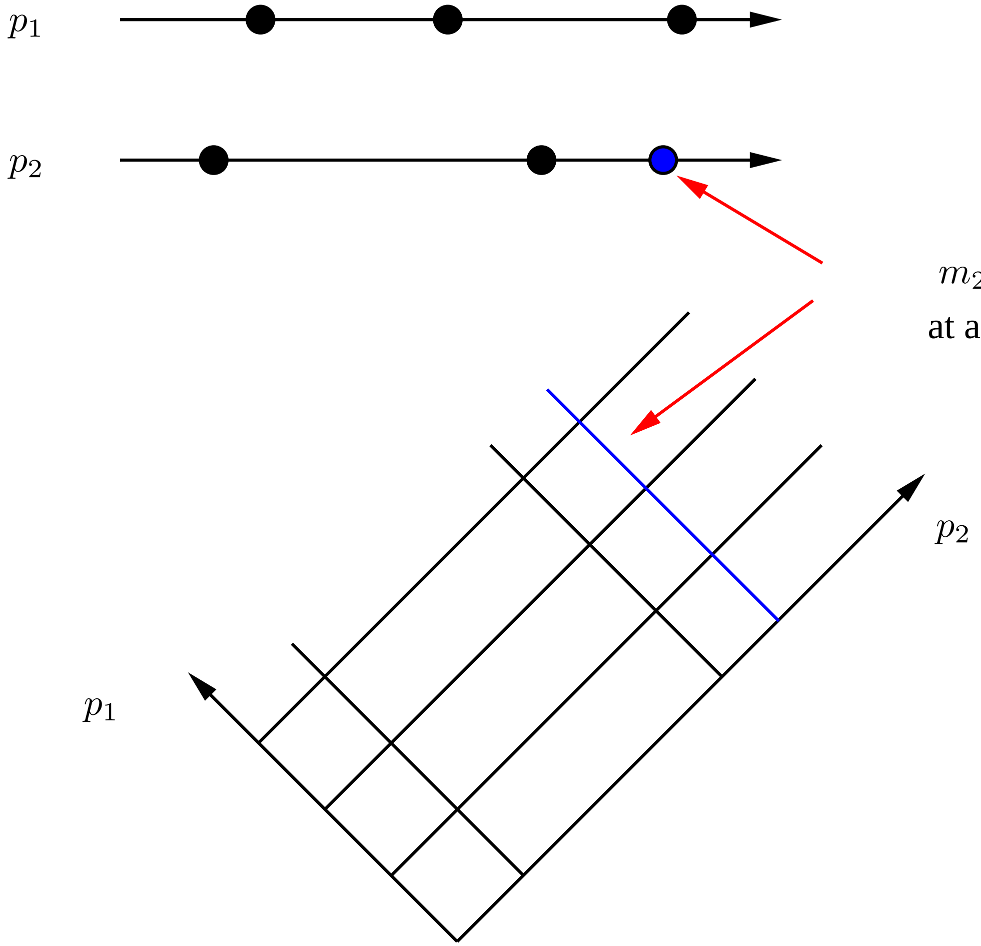
- Let monitors causally broadcast their suspicions to all other monitors.
- Eventually all monitor lattices converge.
- Can then do *possibly*/*definitely* detection in observer invariant state lattices (use standard algorithms).
- Problem: how know that there will be no "late" failure detector events arriving?
- Solution:
  - Monitors piggyback coordinates of most recent global state they have seen: per monitor stable region.
  - Take intersection of all monitor regions: globally settled region.
  - Steadily expand settled region, extract optimistic/pessimistic data and do *possibly*/*definitely* detection on it.

Settled region example



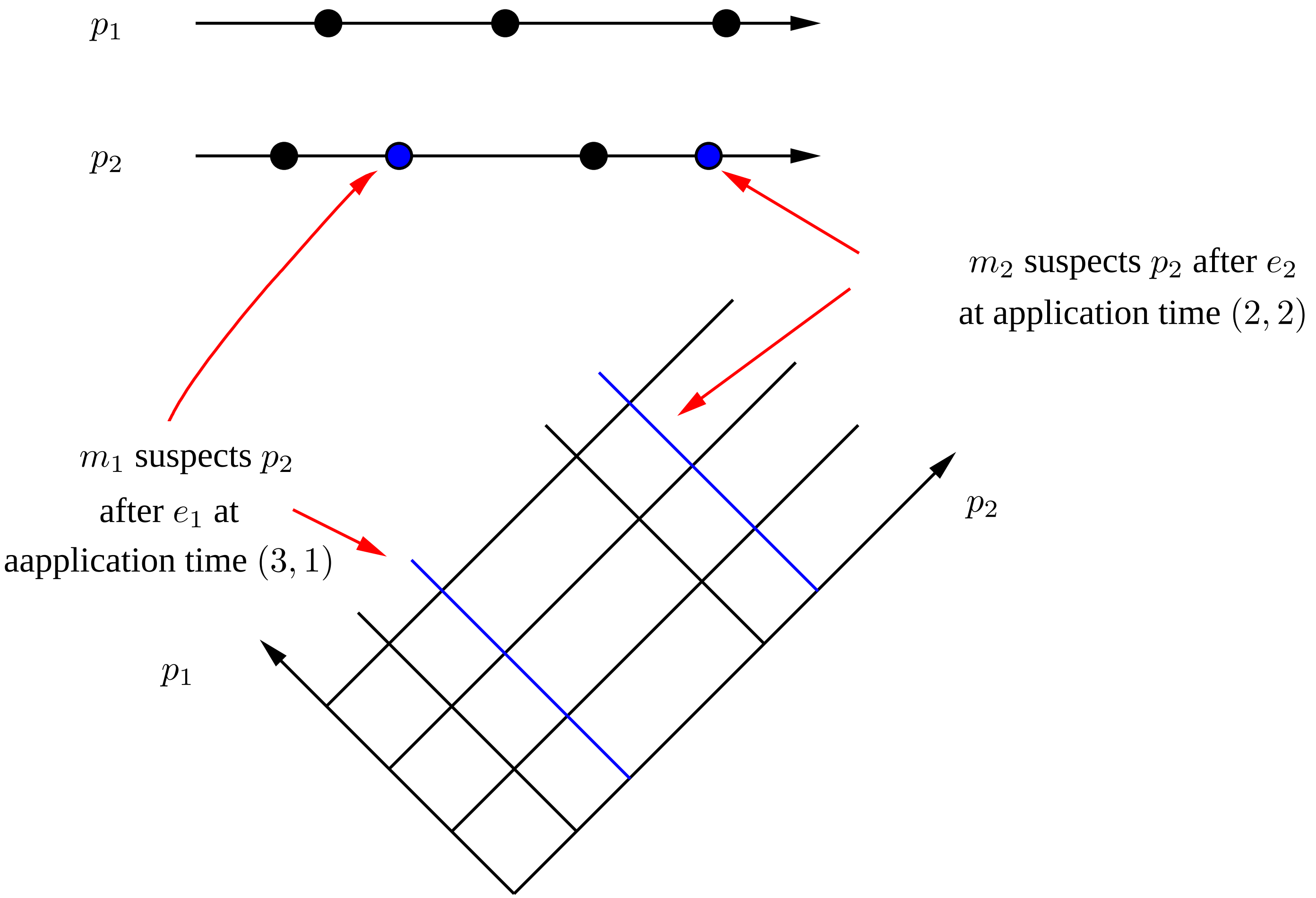


Settled region example



m_2 suspects p_2 after e_2
at application time $(2, 2)$

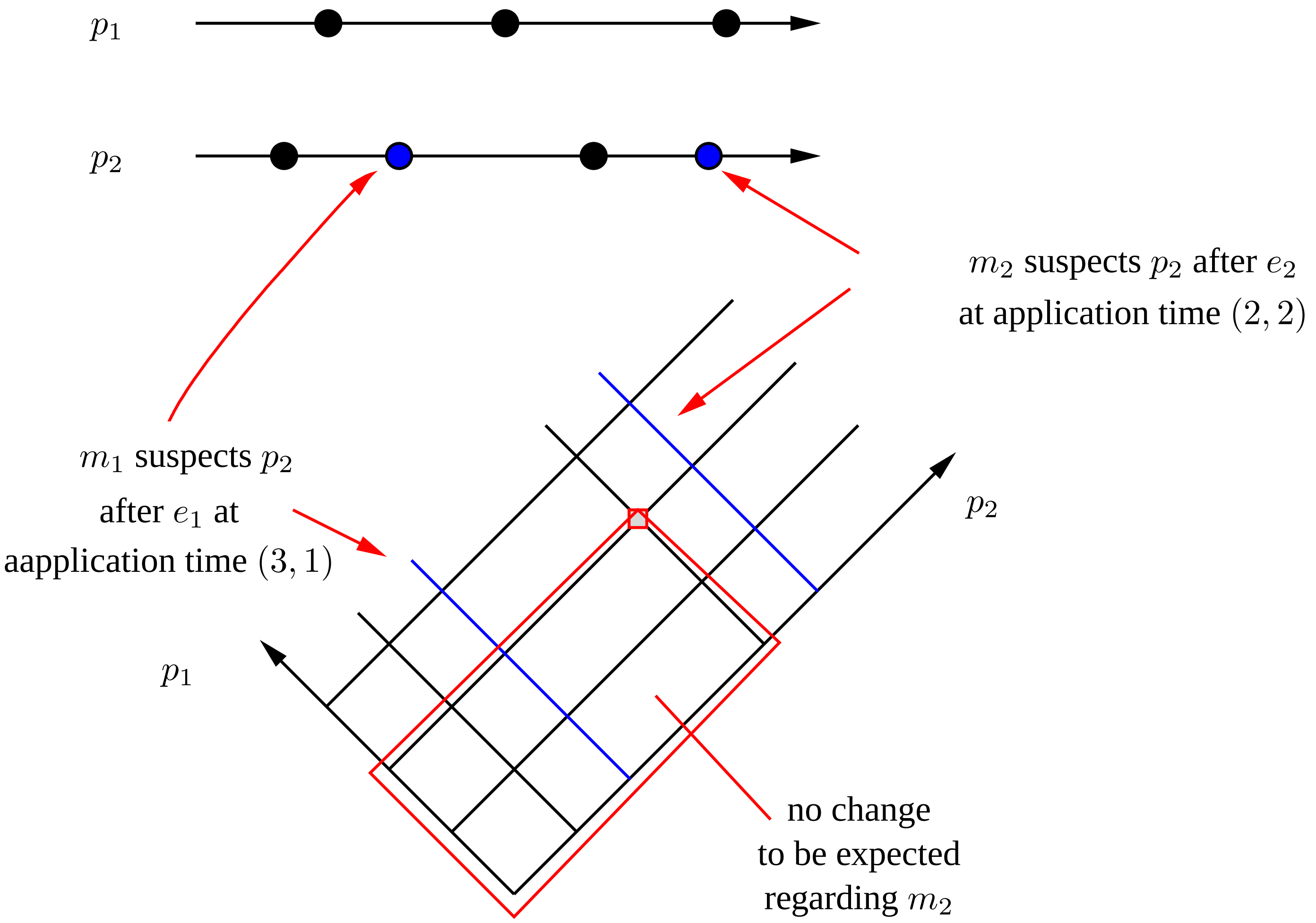
Settled region example



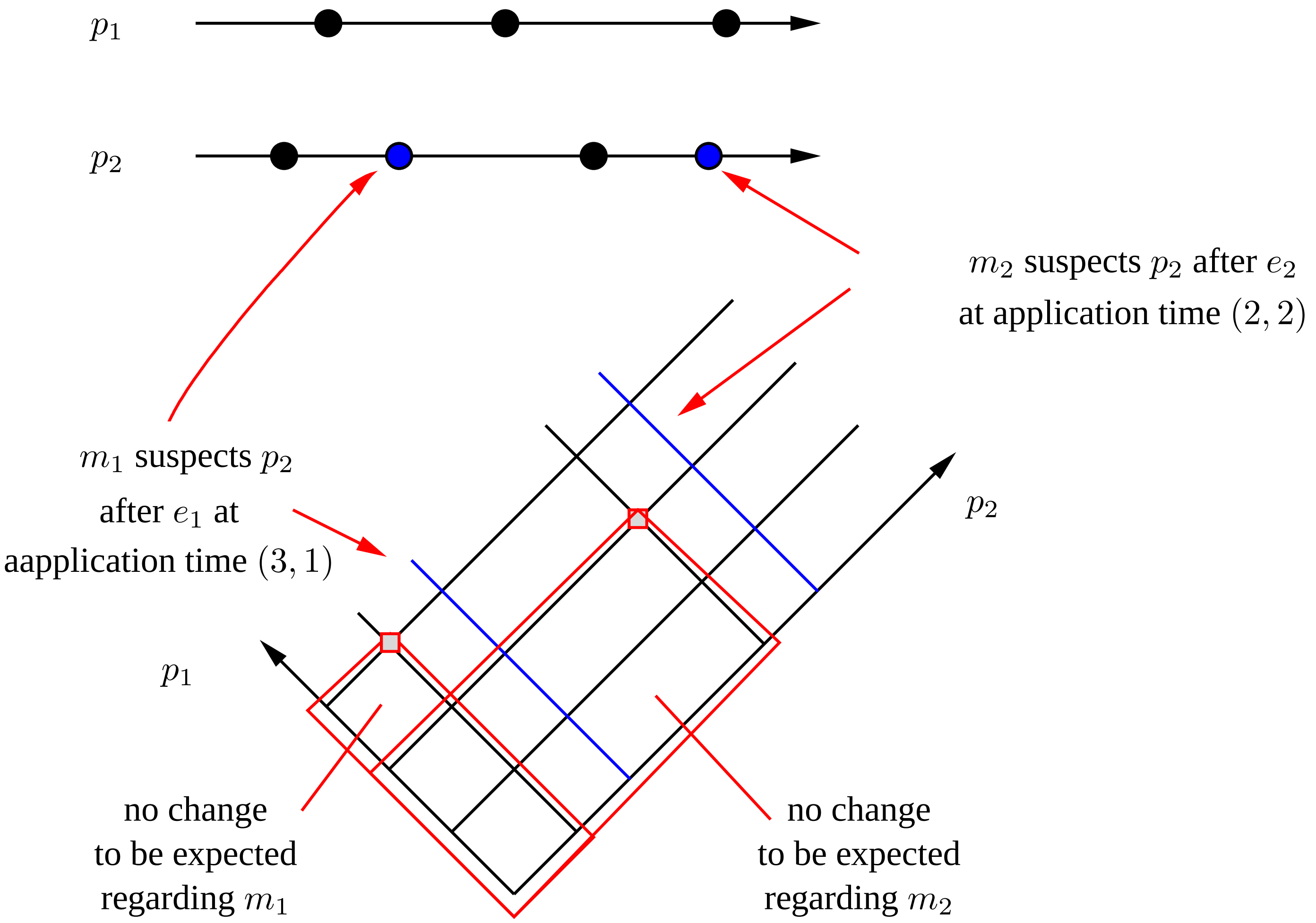
m_2 suspects p_2 after e_2 at application time $(2, 2)$

m_1 suspects p_2 after e_1 at aapplication time $(3, 1)$

Settled region example



Settled region example



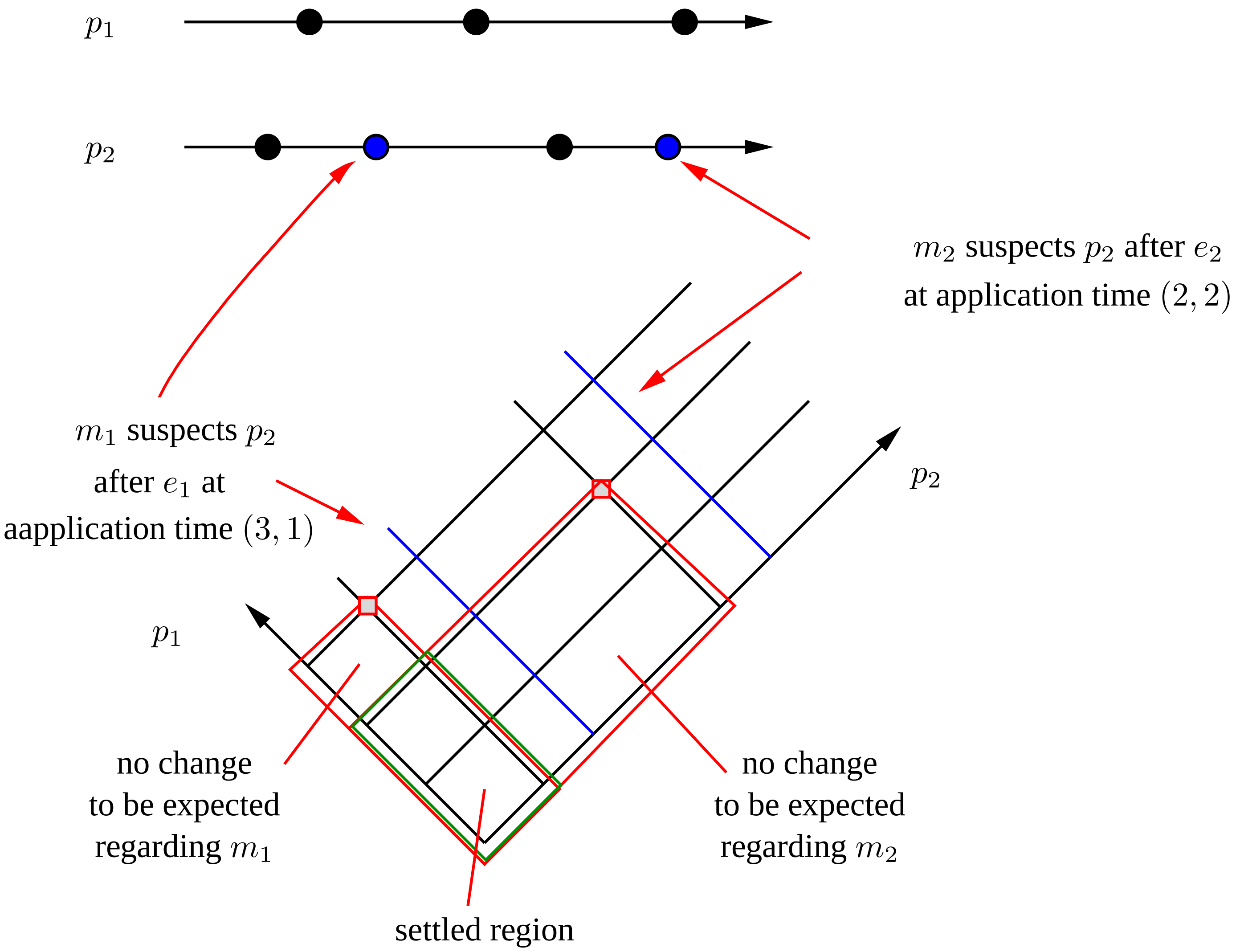
m_2 suspects p_2 after e_2 at application time $(2, 2)$

m_1 suspects p_2 after e_1 at aapplication time $(3, 1)$

no change to be expected regarding m_1

no change to be expected regarding m_2

Settled region example



Advanced topics

- Algorithm works under assumption that no monitors fail.
- If monitors can fail, detection becomes harder:
  - Can still detect $negotiably$ without a stable region.
  - Detection $discernibly$ impossible, because accurate failure detection is needed.
  - A weaker variant (t-$discernably$) can be detected at the price of having a majority of correct monitors.

Complexity and restricted predicates

- Complexity:
  - general predicate detection is NP-complete [1].
  - Our detection algorithms are only wrappers around possibility/definitely detection.
  - Study restricted classes of predicates.
- Perfect failure detectors available:
  - No false suspicions.
  - Optimistic/pessimistic lattice are the same.
- Perfect failure detectors and crash predicates:
  - Predicates are stable.
  - $possibly = definitely \rightarrow negotiably = discernibly$

Overview of results

- First work to deal with general predicates in faulty systems (only other work by Garg and Mitchell [2] restricts the classes of predicates).

- Observation modalities $negotiably$ and $discernibly$. . .
  - do not solve all problems in crash-affected systems.
  - reflect by their definition the inherent problem of crash failure detection.
  - can be understood in analogy to $possibly$ and $definitely$.
  - can be detected in asynchronous systems, even if monitors may crash.

- Still a lot of work to do.

References

[1] Craig M. Chase and Vijay K. Garg. Detection of global predicates: Techniques and their limitations. *Distributed Computing*, 11(4):191–201, 1998.

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

[3] Vijay K. Garg and J. Roger Mitchell. Implementable failure detectors in asynchronous systems. In *Proc. 18th Conference on Foundations of Software Technology and Theoretical Computer Science*, number 1530 in Lecture Notes in Computer Science, Chennai, India, December 1998. Springer-Verlag.

Acknowledgements

- Slides produced using “cutting edge” LaTeX slide processor PPower4 by Klaus Guntermann.