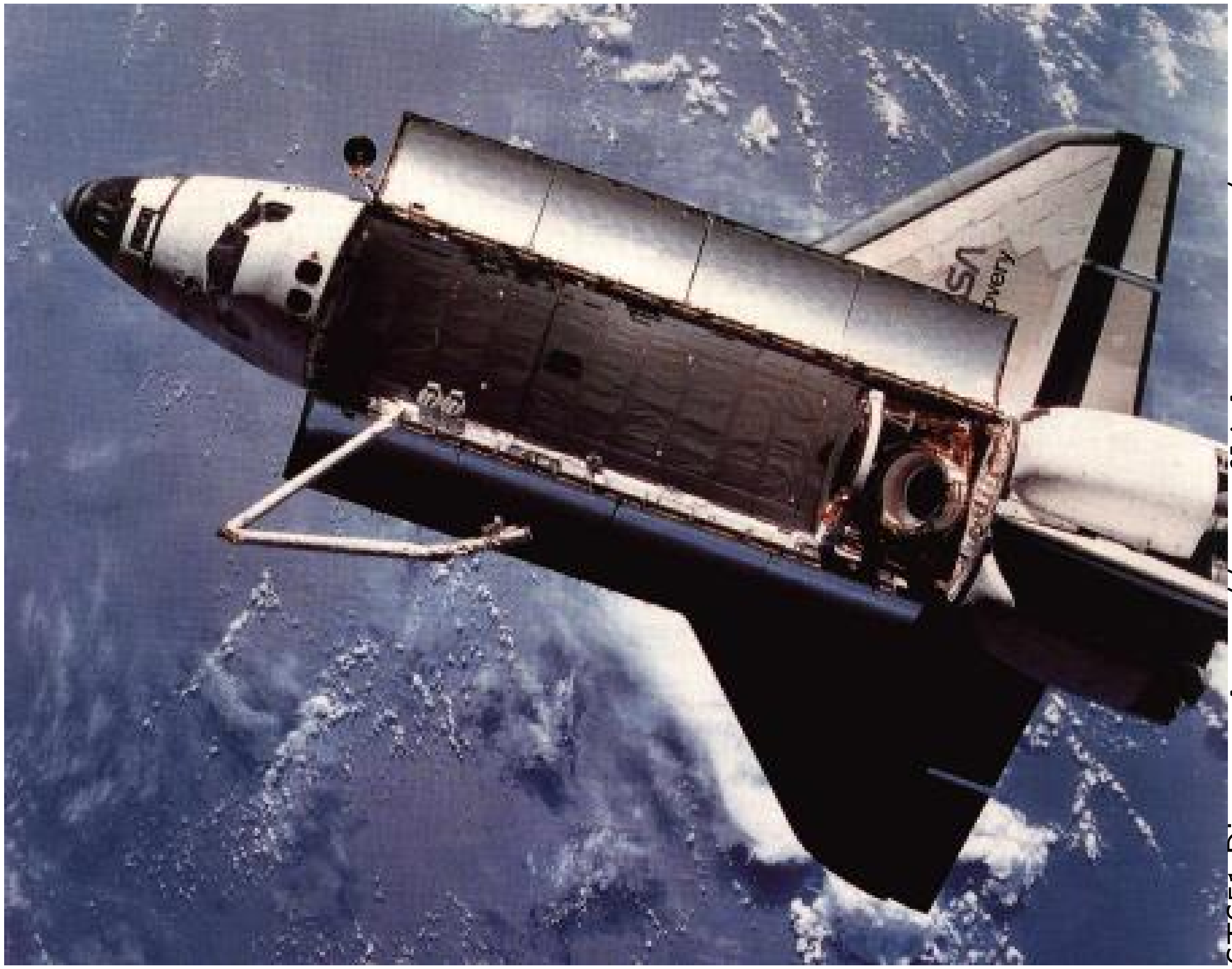
Formalization and Verification of Fault Tolerance and Security

Felix Gärtner

TU Darmstadt, Germany

fcg@acm.org

Example: Space Shuttle



STS51 Discovery, http://spaceflight.nasa.gov/

Fault-tolerant Operation [Spector and Gifford 1984]

- Five redundant general purpose computers.
- Four of them run the avionics software in parallel.
- Majority vote of computation results.
- “Fail-operational, fail-safe.”
- Fifth computer runs backup system (written by separate contractor). Primary contractor: IBM.

Critical Infrastructures

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

- Critical infrastructures must be dependable (in this talk meaning fault-tolerant and secure).

Personal Motivation

- My . . .
  - background: fault-tolerance, formal methods.
  - experience: formal methods help find bugs.
  - concern: need to formalize issues first (state what we mean).
  - claim: we know how to do this in fault-tolerance, not so much in security.

Overview

1. Fault tolerance (60% of talk).
   - What does "fault tolerance" mean?
   - How can it be formalized and verified?
2. Security (30%).
   - What does "security" mean and how can it be formalized???

Informal View of Fault Tolerance

- Definition: Maintain some form of correct behavior in the presence of faults.
- Correct behavior: specification.
- Faults:
  - memory perturbation (cosmic rays),
  - link failure (construction works),
  - node crash (power outage),
  - . . .

Formal View of Fault Tolerance

- System: state machine/event system with interface.
- Specification: look at functional properties defined on individual executions of the system.
- Safety properties: "always . . . ".
- Liveness properties: "eventually . . . ".
- Abstract away from real time.

Safety and Liveness

- Safety properties: observable in finite time.
- Examples: mutual exclusion, partial correctness.
- Liveness property: violated after infinite time.
- Example: starvation freedom, termination.
- Safety and liveness are fundamental [Alpern and Schneider 1985; Gärtner 1999a].

Faults. . .

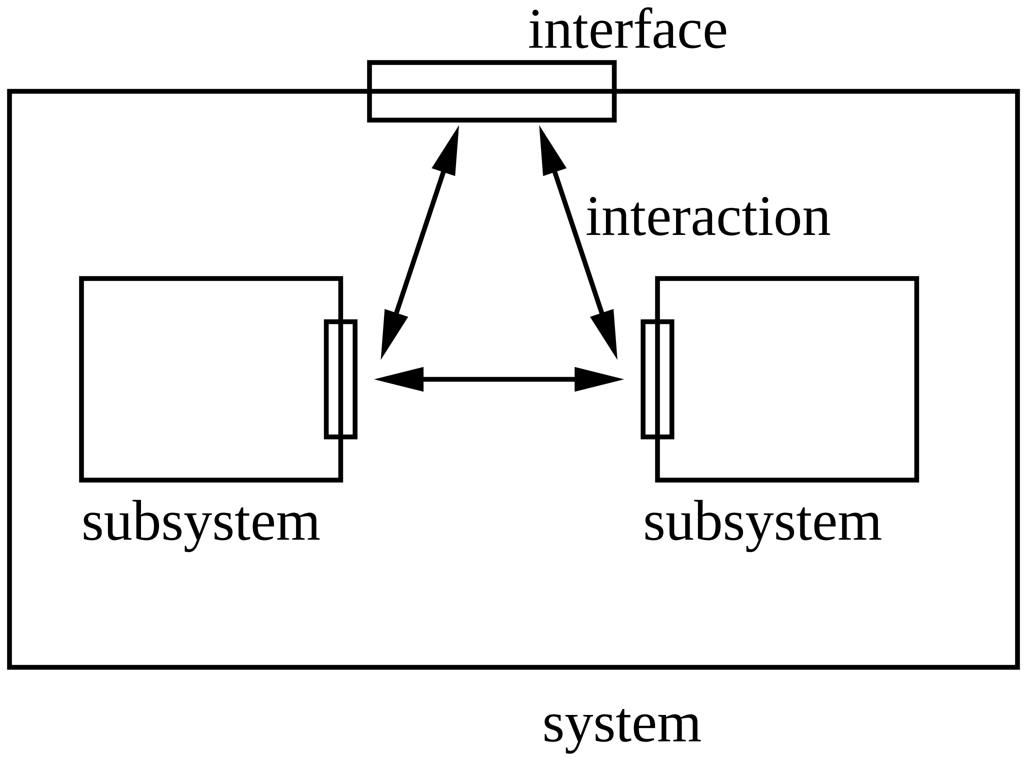
- can be modelled as unexpected events [Cristian 1985].
- are tied to one level of abstraction [Liu and Joseph 1992].
- Adding and “removing” state transitions is enough [Gärtner 2001a].
- are formalized as a fault assumption.

Fault Tolerance Example

- Network of workstations with point-to-point links.
- Fault assumption: links and workstations can crash, but network stays connnected.
- We want to do reliable broadcast.
- Specification (desired properties):
  - A message which is delivered was previously broadcast (safety).
  - A broadcast message is eventually delivered on all surviving machines (liveness).

Fault on one Level of Abstraction

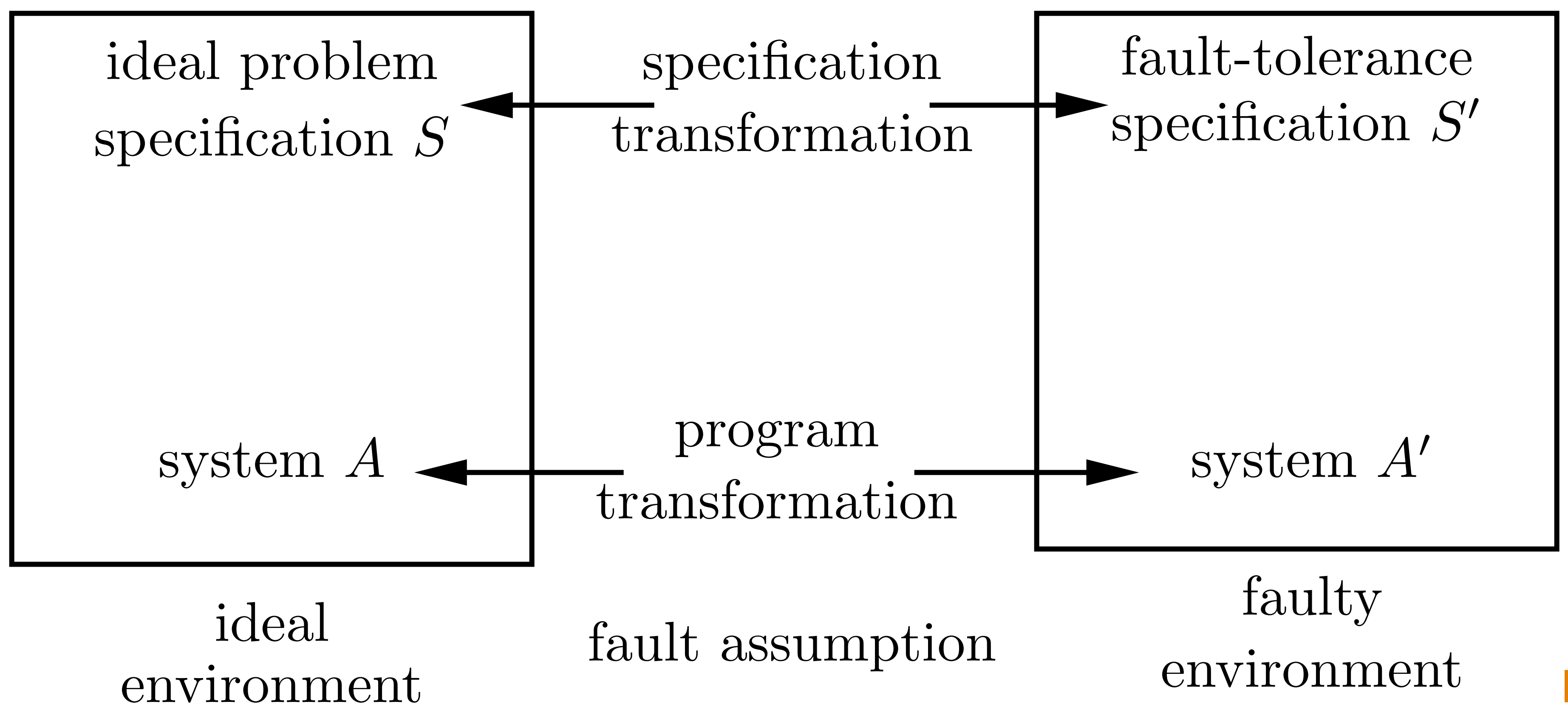
- System = composition of systems.



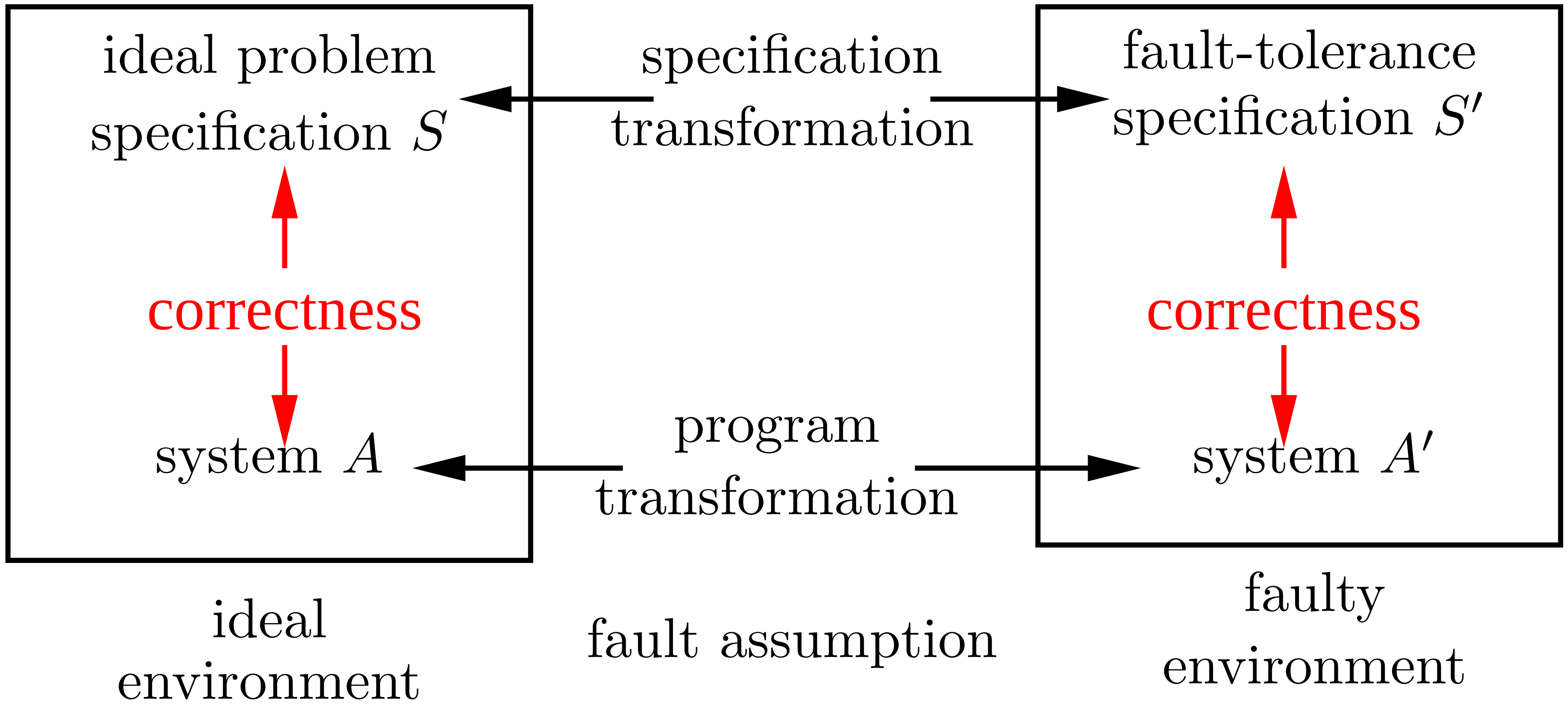
Local and Global Fault Assumptions

- Local fault assumption: add behavior to fault regions.
- Example: node crash allows processes to stop.
- Global fault assumption: restrict behavior again.
- Example: network stays connected.

Fault Assumptions as Transformations [Gärtner 1998]



Verification



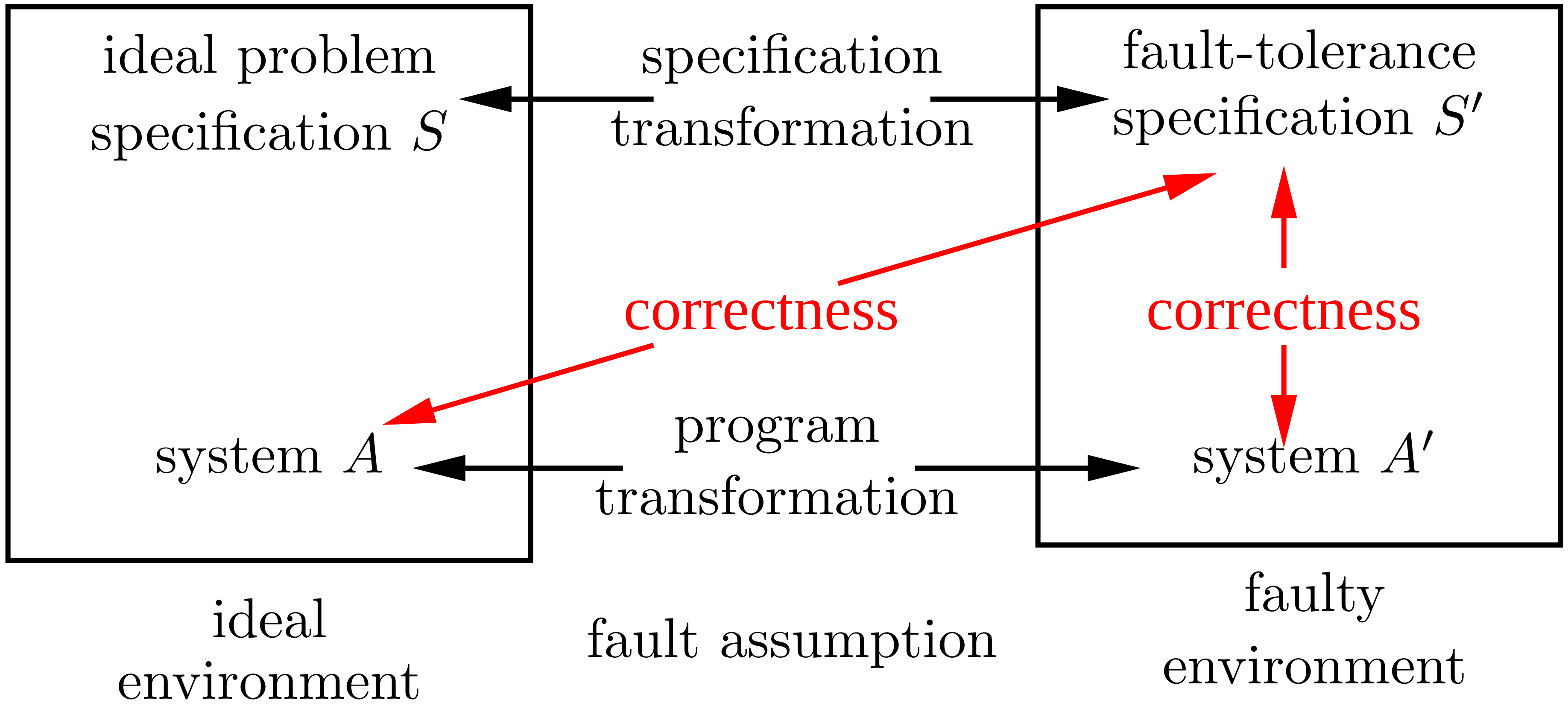
Usual Verification of Fault Tolerance

1. Choose fault assumption.
2. Weaken specification (if needed).
3. Transform system.
4. Verify system.

Transformational Approach [Gärtner 1999b]

1. Choose fault assumption.
2. Weaken specification (if needed).
3. Prove that original system satisfies specification.
4. Transform system.
5. Prove only items which have changed (use tools like VSE, PVS, . . .).

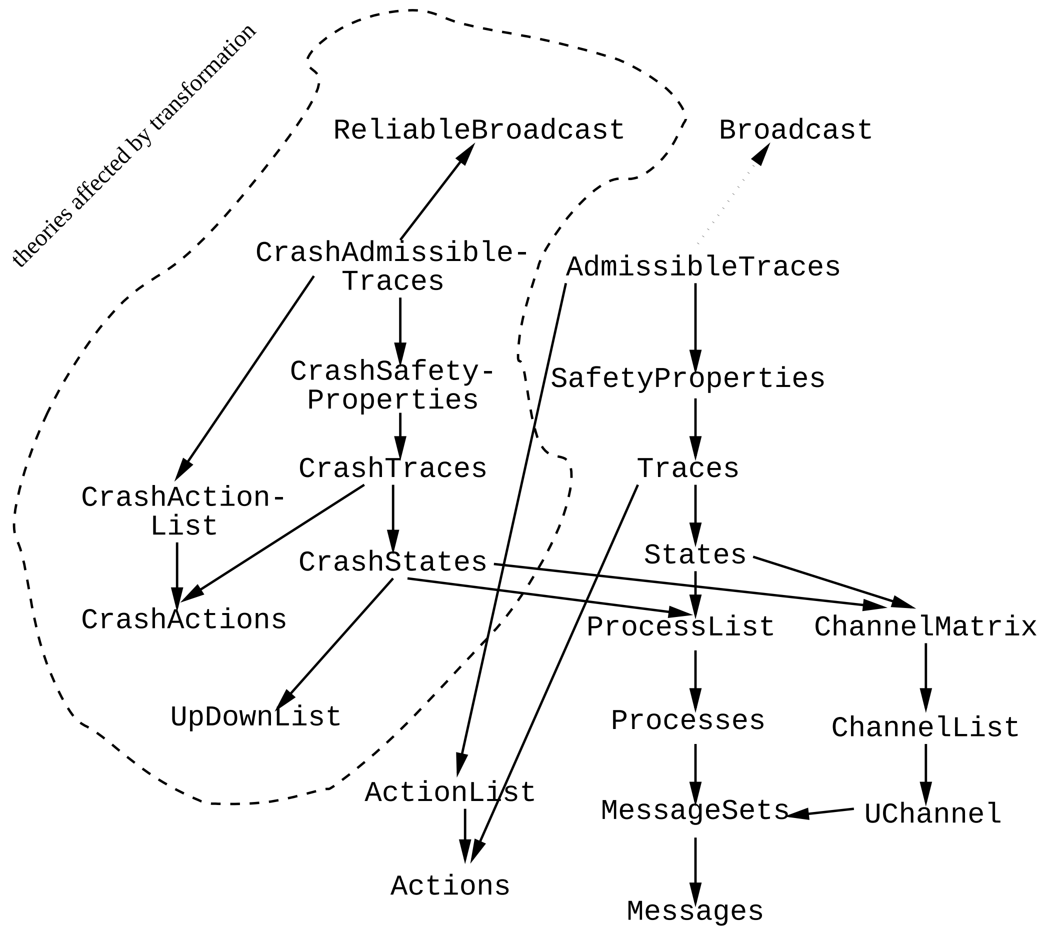
Potential of Re-Use



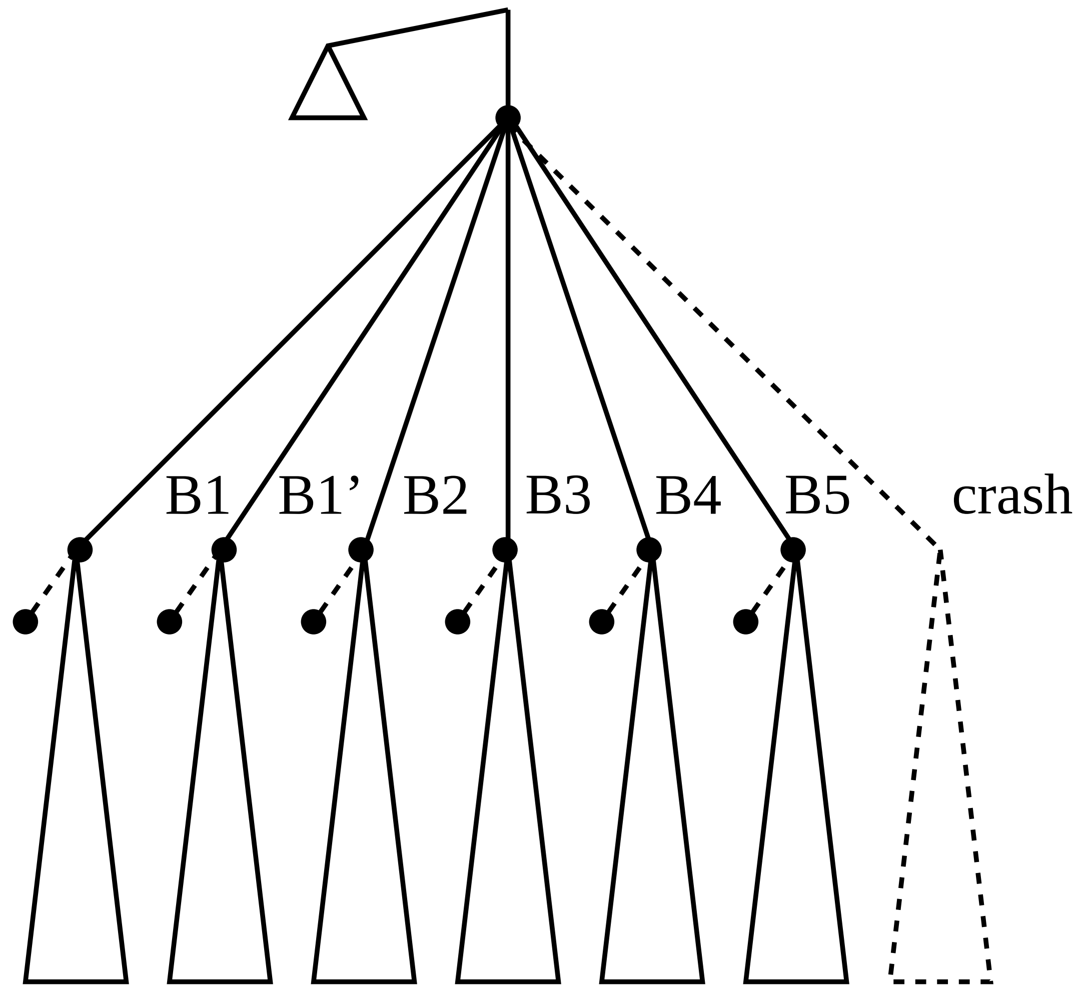
Case Study [Mantel and Gärtner 2000]

- Example: reliable broadcast.
- Proved safety part using industrial strength verification tool VSE [Hutter et al. 1996].
- Transformational approach applied.
- Benefit: re-use of specification and proofs.

Re-use of Specification



Re-use of Proofs



Fault Tolerance Summary

- We basically know how to deal with fault tolerance.
- Formalizations and verification methods are quite mature.
- Area has a solid formal foundation.

Fault Tolerance and Security

- Can research in security benefit from fault tolerance?

"Fault tolerance and security are instances of a more general class of property that constrains influence."

Franklin Webber, BBN (during SRDS2000 panel)

- Example: tolerate malicious behavior by assuming Byzantine faults (like in ISS).

Informal View of Security

- Security is CIA [Laprie 1992]:
  - Confidentiality: non-occurrence of unauthorized disclosure of information.
  - Integrity: non-occurrence of inadequate information alterations.
  - Availability: readiness for usage.
- Conjecture: Everything is CIA! [Cachin et al. 2000]

Formal View of Security

- Recall concepts of safety and liveness (from fault tolerance).
- We can model a lot of notions from security with these concepts, but not all.
- Benefits:
  - Well understood formalisms.
  - Good proof methodologies and tool support.

Safety and Liveness in Security

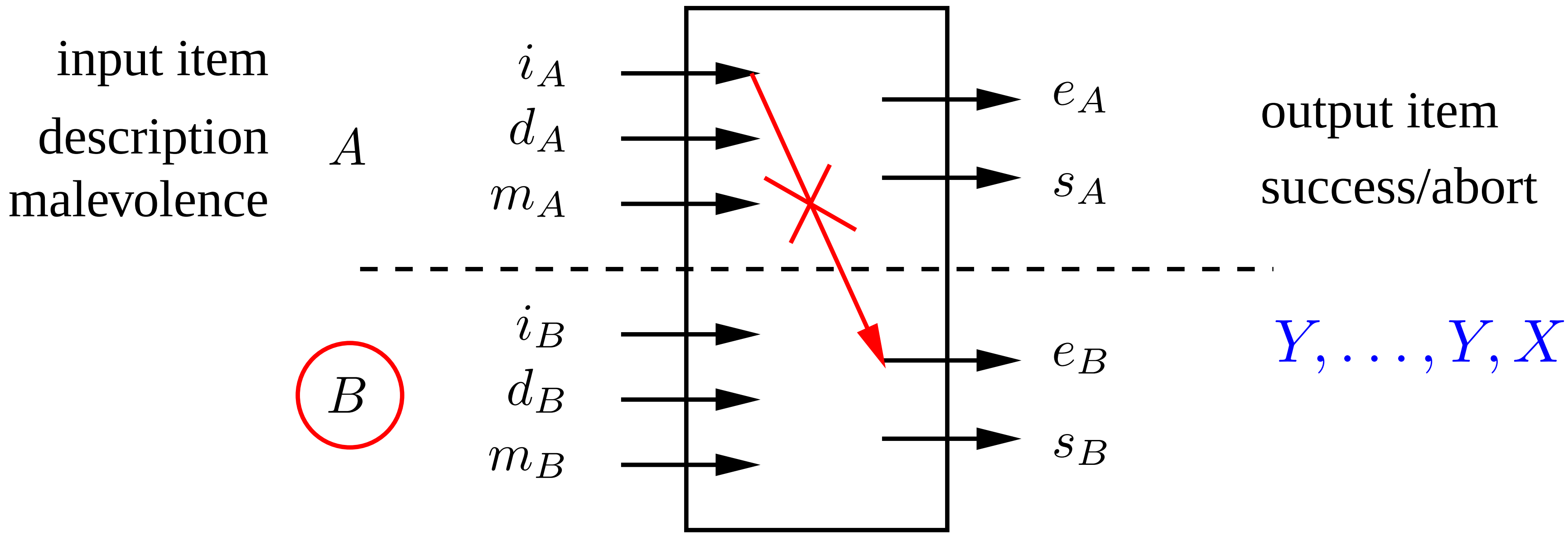
- Access control is safety [Schneider 2000; **?**].
- Aspects of confidentiality are safety [Gray, III. and McLean 1995].
- Aspects of integrity are safety, e.g. "no unauthorized change of a variable".
- Aspects of availability are liveness, e.g. "eventual reply to a request".

Fair Exchange [Asokan et al. 1997]

- Two participants A and B with electronic items.
- How to exchange the items in a fair manner? Formally:
  - Effectiveness: if exchange succeeds then items matched the expectation and both participants have well behaved (safety).
  - Termination: eventually the protocol will terminate with success or abort (liveness).
  - Fairness: in case of an unsuccessful exchange, ange, nobody wins or loses something valuable.

Formalizing Fair Exchange [Gärtner 2001b]

$z, z, z, \ldots$
$x, x, x, \ldots$



$Y, \ldots, Y, X, X, \ldots$

$x, Y, \ldots, x, Y, x, X, x, X, \ldots$
$z, Y, \ldots, z, Y, z, X, z, X, \ldots$

Higher Level Properties

- Consequence: Restriction of information flow is neither safety nor liveness.
- Property of the type: if trace x, X, x, X is possible, then trace z, X, z, X must be possible too.
- Usually formalized as closure conditions on trace sets:

$$\sigma \in S \Rightarrow f(\sigma) \in S$$

- Properties of properties, sets of sets of traces.

Original Approach

- Non-interference [Goguen and Meseguer 1982].
- Descendants with their own problems [McLean 1994]:
  - Generalized non-interference.
  - Restrictiveness.
  - Non-inference.
  - . . .
- Possibilistic properties.

Possibilistic Properties

- Pure non-interference is too strong.
- There is progress in weakening the definition to make it practical [Mantel 2000].
- First results available [Focardi et al. 1997].
- To be investigated: relation to other ways to specify security [Pfitzmann et al. 2000].

Motivation Reminder

- Formal methods are no silver bullet, but they help to find bugs in critical systems.
- Starting point: formalization of central concepts.
- We know how to do that in fault tolerance.
- But fault tolerance seems "easy" compared to security.
- Security defines a new class of properties.

Historic Perspective

"The first wave of attacks is physical [e.g. cut wires]. But these problems we basically know how to solve."

$\longrightarrow$ fault tolerance

The second wave is syntactic [e.g. exploiting vulnerabilities]. We have a bad track record in protecting against syntactic attacks. But at least we know what the problem is.

$\longrightarrow$ security models

Bruce Schneier (Inside Risks, Dec. 2000)

Conclusions 1/2

- We seem to have managed dealing with physical attacks.
- Currently trying to cope with syntactic ones.
- We need a thorough understanding of the concepts involved.
- Formal methods can support rigorous anaysis.
- Formalization is the first step.

Conclusions 2/2

- We've come a long way in formal analysis.
- Milestones: safety, liveness, (linear) temporal logic for modeling functional (trace set) properties.
- Shifting to more difficult properties: security, possibilistic properties.
- Open issue: Is this formalization adequate/useful?
- What about semantic attacks (e.g. stock market hoaxes)?

Acknowledgments

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

References

Alpern, B. and Schneider, F. B. 1985. Defining liveness. *Information Processing Letters 21*, 181–185.

Asokan, N., Schunter, M., and Waidner, M. 1997. Optimistic protocols for fair exchange. In T. Matsumoto Ed., *4th ACM Conference on Computer and Communications Security* (Zurich, Switzerland, April 1997), pp. 8–17. ACM Press.

Cachin, C., Camenisch, J., Dacier, M., Deswarte, Y., Dobson, J., Horne, D., Kursawe, K., Laprie, J.-C., Lebraud, J.-C., Long, D., McCutcheon, T., Müller, J., Petzold, F., Pfitzmann, B., Powell, D., Randell, B., Schunter, M., Shoup, V., Veríssimo, P., Trouessin, G., Stroud, R. J., Waidner, M., and Welch, I. S. 2000. Reference model and use cases. Deliverable D1 of the MAFTIA project [MAFTIA].

Cristian, F. 1985. A rigorous approach to fault-tolerant programming. *IEEE Transactions on Software Engineering 11*, 1 (Jan.), 23–31.

Focardi, R., Ghelli, A., and Gorrieri, R. 1997. Using non interference for the analysis of security protocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols* (DIMACS Center, Rutgers University, Sept. 1997).

Gärtner, F. C. 1998. Specifications for fault tolerance: A comedy of failures. Technical Report TUD-BS-1998-03 (Oct.), Darmstadt University of Technology, Darmstadt, Germany.

Gärtner, F. C. 1999a. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Computing Surveys 31*, 1

(March), 1–26.

Gärtner, F. C. 1999b. Transformational approaches to the specification and verification of fault-tolerant systems: Formal background and classification. *Journal of Universal Computer Science (J.UCS) 5*, 10 (Oct.), 668–692. Special Issue on Dependability Evaluation and Assessment.

Gärtner, F. C. 2001a. *Formale Grundlagen der Fehlertoleranz in verteilten Systemen*. Ph. D. thesis, Fachbereich Informatik, TU Darmstadt. forthcoming.

Gärtner, F. C. 2001b. Formalizing fairness in electronic commerce using possibilistic security properties. Technical report, Darmstadt University of Technology, Department of Computer Science. to appear.

Goguen, J. A. and Meseguer, J. 1982. Security policies and security models. In *Proceedings of the 1982 Symposium on Security and Privacy (SSP '82)* (Los Alamitos, Ca., USA, April 1982), pp. 11–20. IEEE Computer Society Press.

Gray, III., J. W. and McLean, J. 1995. Using temporal logic to specify and verify cryptographic protocols. In *Proceedings of the Eighth Computer Security Foundations Workshop (CSFW '95)* (Washington - Brussels - Tokyo, June 1995), pp. 108–117. IEEE.

Hutter, D., Langenstein, B., Sengler, C., Siekmann, J. H., Stephan, W., and Wolpers, A. 1996. Verification support environment (VSE). *High Integrity Systems 1*, 6, 523–530.

Laprie, J.-C. Ed. 1992. *Dependability: Basic concepts and Terminology*, Volume 5 of *Dependable Computing and Fault-Tolerant Systems*. Springer-Verlag.

Liu, Z. and Joseph, M. 1992. Transformation of programs for fault-tolerance. *Formal Aspects of Computing 4*, 5, 442–469.

MAFTIA. Maftia home – malicious- and accidental-fault tolerance for internet applications. Internet: `http://www.newcastle.research.ec.org/maftia/`.

Mantel, H. 2000. Possibilistic definitions of security - an assembly kit. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop,* (Cambridge, England, July 2000). IEEE Computer Society Press.

Mantel, H. and Gärtner, F. C. 2000. A case study in the mechanical verification of fault tolerance. *Journal of Experimental & Theoretical Artificial Intelligence 12*, 4 (Oct.). to appear.

McLean, J. 1994. Security models. In J. Marciniak Ed., *Encyclopedia of Software Engineering*. John Wiley & Sons.

Pfitzmann, B., Schunter, M., and Waidner, M. 2000. Secure reactive systems. Research Report RZ 3206 (#93252) (Feb.), IBM Research.

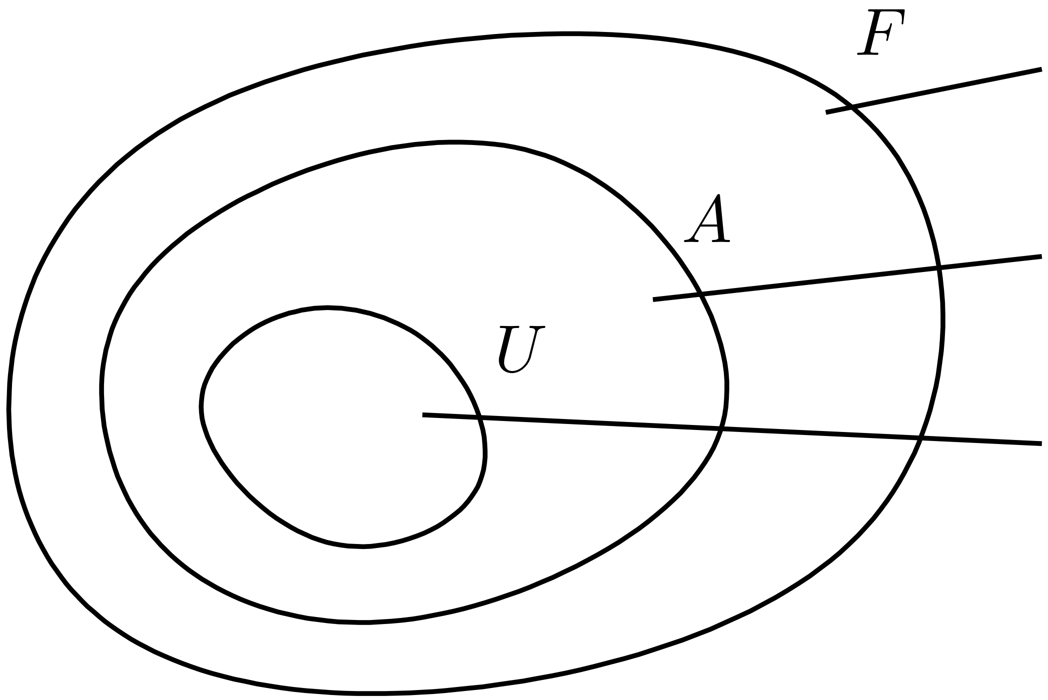
Schneider, F. B. 2000. Enforceable security policies. *ACM Transactions on Information and System Security 3*, 1 (Feb.), 30–50.

Spector, A. and Gifford, D. 1984. The space shuttle primary computer system. *Communications of the ACM 27*, 9, 874–900.

Abstract

It is often argued that fault tolerance and security are similar properties and can be achieved by similar means. In this talk I will first give an overview of methods used to formalize fault tolerance, especially those aimed at verification and validation of fault-tolerant systems, and briefly present a case study in which these methods have been successfully applied. In the remaining part of the talk, I will sketch different ways how security properties have been formalized and how experience from fault tolerance can help in the clarification of the issues involved. It turns out that while some aspects of security are in fact closely related to fault tolerance, other aspects (like confidentiality) are fundamentally different in nature. To initiate discussion, I will speculate on promising ways of how to deal with these issues from a practicioner's point fo view.

Appendix: Proof that Fairness is not a Trace Set Property



assumed trace set
absence of info flow

set of all unsuccessful traces is fair

restriction of A to all traces that give away the item