

1 Project info

1a) **Project Title:** Fault-tolerant Real-time Algorithms Analyzed Incrementally

1b) **Project Acronym:** FRAAI

1c) **Principal Investigator:** prof.dr. F.W. Vaandrager

2 Summary

In many application domains there is a strong dependency on computing systems and the availability of a continuous service is often extremely important. Most of these systems are designed to be fault-tolerant.

The enormous complexity of the algorithms that achieve fault-tolerance makes it impossible to obtain sufficient confidence in their correctness just by informal inspection. Formal methods have been applied successfully to validate smaller examples, but it is evident that more research is needed to be able to handle large examples.

In practice, typically an *incremental* approach is followed. One starts with a (relatively) simple algorithmic problem, for instance including only some simple failures, and, after a solution for this problem has been verified, one gradually (1) weakens the failure hypothesis, and/or (2) strengthens the specification by adding requirements. The incremental approach is often quite successful because the same arguments that ensure correctness of the basic algorithm remain (more or less) valid for the more complex versions. The aim of this project is to formalize this type of reasoning: we want to develop theory which allows us to reuse formal correctness proofs of simple fault-tolerant algorithms within proofs of more complex ones.

3 Classification

6.5, 1.3, 1.2

4 Composition of the Research Team

The next table specifies the persons directly involved in this project, which will be carried out within the group Informatics for Technical Applications (see <http://www.cs.kun.nl/ita/>) at the University of Nijmegen (KUN), the Netherlands. Prof. Vaandrager will act as promotor of the PhD student.

Name	Specialism	hrs/week
prof.dr. F.W. Vaandrager	automata based verification, embedded systems	4
dr. J.J.M. Hooman	assertional verification, PVS	2
drs. H.C.W. Kuppens	scientific programmer	8
NN (Project PhD student)	protocol verification	40

5 Research School

The research in this project will be carried out in the context of research school IPA (Institute for Programming research and Algorithmics).

6 Description of the Proposed Research

6.1 Problem Statement

In many application domains there is a strong dependency on computing systems and the availability of a continuous service is extremely important. Examples include process control, aircraft control, telecom applications and on-line transaction processing (e.g. stock exchange). Failures of hardware and software components are, however, unavoidable. Hence most of these systems are designed to be fault-tolerant. For instance, special algorithms are used to detect errors, minimize loss of services due to failures, and to recover from component failures.

In general, there is a hierarchy of distributed algorithms that provide a certain real-time and fault-tolerant service (see, e.g. [Cri89b]). For instance, starting from a network of processors with local clocks and a simple communication mechanism, there are distributed algorithms that ensure synchronized clocks. Based on that, a more powerful communication mechanism, such a atomic broadcast, can be implemented. This mechanism can then be used to implement certain agreement algorithms, e.g. to reach a consistent view on the set of correct processors which can be used to redistribute workload.

Since such a stack of algorithms/protocols may form the basis for numerous critical applications, correctness is often vitally important. However, the combination of distribution, local clocks, real-time, and fault-tolerance makes it extremely difficult to guarantee that these algorithms indeed provide to the required services. Observe that real-time and fault-tolerance are intertwined; timing information is often used to detect errors and redundancy introduced for fault-tolerance influences the real-time behavior of the system. Moreover, when recovery from failures is important, one also has to reason about different types of memory; volatile memory has fast access but might get lost during crashes, so an essential part of the state has to kept in stable memory.

This complexity makes it impossible to obtain sufficient confidence in the correctness of these algorithms just by informal inspection, especially due to the possibility of the combination of failures. Formal methods have been applied successfully to validate smaller examples. But even though in the literature various methods have been proposed to break down the verification task for these protocols into manageable subtasks (stepwise refinement, compositionality, communication closed layers, abstract interpretation,..) and even though there has been enormous progress in the area of tool support (model checkers, theorem provers,..) it is evident that more research is needed to be able to formally verify these large and complex algorithms.

6.2 Approach

The proposed project aims to address the above problem along three lines:

1. **Theory**

In particular, we aim at results that permit *incremental* verification of fault-tolerant algorithms (or, more generally, reactive systems).

2. **Tools**

Strong tool support is indispensable when dealing with complex systems. We emphasize the use of theorem provers, although model checkers and simulation tools will play a valuable role in the exploration phase.

3. **Applications**

Since scalability of the proposed method is a key issue, we will evaluate our ideas on a number of complex fault-tolerant algorithms which also include some recovery possibilities.

Below, we will elaborate on these three research lines.

6.2.1 Theory

Complex fault-tolerant algorithms and protocols are typically structured in terms of a number of layers, built on top of each other as for instance in the ISO OSI protocol hierarchy [Tan81].

Within each individual layer, one can often identify a number of sequential phases. Over the last two decades, much theory has been developed which allows one to exploit this type of overall structure of reactive systems in the process of formal verification. Roughly speaking, the theory of stepwise refinement [AL91, LV95, RE98] allows one to deal with each layer separately, whereas the theory of communication closed layers [JPXZ94] makes it possible to analyze separately the phases within each layer. But even though in this way one can break down the huge complexity of realistic fault-tolerant systems, the complexity of the remaining parts is often still enormous.

In practice, typically an *incremental* approach is followed in both the design and verification of complex distributed algorithms. One starts with a (relatively) simple algorithmic problem, for instance including none or only some simple failures, and, after a solution for this problem has been designed and verified, one gradually (1) weakens the failure hypothesis, and/or (2) strengthens the specification of the algorithm by adding requirements. Thus, for example, one may attempt to deal with additional classes of failures, establish graceful degradation in the case of too many failures, increase performance, consider additional protocol services such as the detection of undesired loops in the network topology, etcetera. The incremental approach is often quite successful because the same arguments that ensure correctness of the basic algorithm remain (more or less) valid for the more complex versions. The aim of the proposed project is to formalize this type of reasoning: we want to develop theory which allows us to reuse formal correctness proofs of simple algorithms within proofs of more complex algorithms.

Basically, we have the following picture:

$$\begin{array}{ccc}
 \textit{AlgorithmA} & \sqsubset_1 & \textit{SpecificationA} \\
 & \sqcup_2 & \sqcup_3 \\
 \textit{AlgorithmB} & \sqsubset_4 & \textit{SpecificationB}
 \end{array}$$

If both algorithms and specifications are represented as automata then \sqsubset_1 and \sqsubset_4 denote behavior inclusion; this approach is for instance taken in the I/O automata framework [LT87, Lyn96]. If both algorithms and specifications are represented as logic formulas, then \sqsubset_1 and \sqsubset_4 simply denote logical implication; this is the TLA approach [Lam94, Lam99]. Finally, if algorithms are represented as automata and specifications as logic formulas, then \sqsubset_1 and \sqsubset_4 correspond to logical satisfaction (\models); this is for instance the approach of Manna and Pnueli [MP92, BBC⁺00]. Depending on the types of their arguments, also relations \sqsubset_2 and \sqsubset_3 may correspond to behavior inclusion, logical implication or satisfaction, but this is not necessary. If, for instance, *AlgorithmB* is obtained from *AlgorithmA* by introducing more failures, then actions of *AlgorithmB* may be executed several times only partially and we have no classical refinement. Also, in the case of new failures it is often allowed to maintain a weaker service. In such a case there will be no refinement relation between *SpecificationB* and *SpecificationA*. Still, our goal will always be to reuse the proofs of \sqsubset_1 , \sqsubset_2 and \sqsubset_3 in the proof of relation \sqsubset_4 . Possible ways to achieve this are

- Formulate restrictions under which extensions remain to satisfy the original spec (e.g. because new failures are “undone” such that a good state is obtained).
- Find a suitable reformulation of refinement, e.g. by making a distinction between correct actions (that should correspond to actions of earlier levels) and actions from which one has to recover and shown that their effect has been undone.
- Represent proofs in a clear and inspectable way such that it is easy to check where disturbances can be expected (and repaired) by new failures. A proof representation which is particularly promising is the notion of a (generalized) verification diagram [MP95, Lam95, MBSU98, BBC⁺00].

Another theoretical question that we would like to address within the project, less important but still worth mentioning, concerns the choice of suitable representations of the algorithms. Here several points are relevant. First one wants to choose a representation close to the informal formulation. E.g. protocols are often represented in some form of pseudo code or as a state machine

(automaton), whereas properties of communication media of the effect of failures are often more closely represented by some logical assertion. To get more confidence in the assertional spec, it might be beneficial to formulate an equivalent automaton, thus also establishing consistency. For verification purposes, it might be convenient to switch from one representation to the other. In general, theoretical results indicate that it should be possible to move from one representation to the other but that in different situations the one or the other might be more concise. Concerning the representation, we would like to combine our mutual expertise in the field of protocol verification, combining assertional and automata approaches. Note that both approaches are state-based.

6.2.2 Tools

As pointed out by Wolper in his inaugural speech, manual verification is at least as likely to be wrong as the program (or algorithm) itself. To manage the complexity of verification, tool support is therefore indispensable. Different tools are needed at various places of the verification process. Even though it is our explicit aim to use existing tools as much as possible, some work will have to be done to link existing tools to each other, to extend these tools with new features, and to write interfaces to make the interaction with the tools more efficient. Based on our experience within the VHS project, see for instance [BHV00, BFH⁺01b, BFH⁺01a, HRSV01], we believe that in order to get the required work done, 0.2 fte support of an experienced scientific programmer is required. The research related to tools will mainly be a matter of engineering: find the most suitable combination of existing tools for different phases of the verification. Below we list a number of different classes of tools that we will probably use during different stages of the project.

Model checking If a system is finite-state, arbitrary temporal properties can be automatically established or refuted, using explicit-state or symbolic model checking [CGP99]. Even though distributed fault-tolerant real-time algorithms typically are highly parameterized and infinite state, we expect that it will be useful, during the so-called exploration phase, to construct finite-state instances of these algorithms that can be model checked to search for bugs, and that can be used for simulation. This will help in getting more confidence in the understanding and modeling of the dependable systems. Within our group we have extensive experience with a number of model checkers, such as Uppaal [BGK⁺96], HyTech [HH95] and Spin [Hol91].

Deductive verification After the exploration phase we move to more realistic models with larger data-dependency (e.g. for recovery there are often rather complex data structures in stable memory), and intend to use theorem-proving. For various reasons (support of higher-order logic, powerful prover,...) we will probably use PVS [ORSH95].

Verification diagrams Verification diagrams provide a high-level, graphical representation of the proof of a (temporal) property [MP95, Lam95, MBSU98, BBC⁺00]. By means of an algorithmic procedure it can be checked that the diagram indeed proves the property at hand. Verification diagrams have been implemented very nicely in the Stanford Temporal Prover STeP [BBC⁺00]. Although we will certainly benefit a lot from the work in STeP, we are not planning to use this tool in the main case studies of our project, mainly because (a) the assertion language of STeP is based in first-order logic; we believe that in order to deal with complex fault-tolerant distributed algorithms we need the expressivity and flexibility of higher-order logic, and (b) thus far, STeP does not support stepwise refinement; we believe stepwise refinement is essential in an incremental approach to verification. A consequence is that we will have to reimplement much of the work on verification diagrams in the setting of PVS (possibly including an interface that can handle verification diagrams). A difference is that we plan to develop verification diagrams for refinement proofs, whereas STeP supports verification diagrams for proofs of temporal logic formulas.

Invariant generation Recently, some very promising techniques have been proposed by which simple invariants of the system being analyzed can be generated automatically [BLS96, BBM97].

We hope to be able to use the package for invariant generation that is currently implemented on top of PVS at the University of Kiel, based on the approach of [BLS96].

6.2.3 Applications

Since scalability of the proposed approach is an important issue, we intend to evaluate our ideas on an industrial fault-tolerant system. Although it is usually difficult to get a realistic case study which is still interesting for industry, there are excellent opportunities to obtain such an example in the current project. Prof.dr. Jeroen Bruijning of KPN Research (and part-time affiliated with the University of Nijmegen), already expressed his strong interest in our project and proposed an interesting case study. This concerns the TAPS/NODE distributed platform for high-speed low cost transaction processing that recently became operational at PTT Telecom. Low costs are partly realized by using a cheap platform, consisting of standard PC's and a local area network. However, such components impose strong requirements on the software to guarantee the expected behavior despite failures of components. In the current system there is a task manager which coordinates the tasks in the system, making sure that all data is correctly processed and recorded. Fault-tolerance is insured by a number of measures, such as some two-phase-commit protocol, a heartbeat algorithm to detect malfunctioning processors, and the use of checkpoints on stable storage to be able to continue after recovery without the loss of much data. There is also a protocol for the atomic update of a table with task information at the coordinator. Note that also the coordinator might crash and suitable data has to be stored on stable memory to allow fast recovery.

In addition to this large case study, we intend to study other individual algorithms that involve fault-tolerance, especially recovery protocols. There are many sources for such protocols, see for instance [Mul93] and the protocols of Cristian [Cri89a, Cri91, CASD95, Cri96]. Precise descriptions of interesting protocols can also be found in [LMWF94, Lyn96]. Another source for interesting algorithms is an extensive bibliography of fault-tolerant distributed computing by Coan [Coa90].

6.3 Related Work

An early example of a design approach allowing for “invalid” intermediate stages can be found in the work of Dijkstra [Dij79] (for a more formal reconstruction, see [CKdR92]). A recent example of an incremental design approach that also allows for invalid intermediate stages is [MBWB01], but in this work no attempt is made to reuse correctness proofs.

An early formal approach to the design of fault-tolerant systems can be found in the work of Schlichting and Schneider [SS83], who propose an axiomatic program verification technique for developing provably correct programs for fail-stop processors. Also Cristian [Cri84, Cri85] presents an axiomatic approach based on extended Hoare triples to deal with the effects of faults. A Hoare-style proof system for CSP-like programs with failures can be found in [JMS87]. In [LJ92] a fault-free program is transformed into a fault-tolerant program, by applying a fault-transformation (introducing failures) and a subsequent recovery transformation. This transformational approach is applied to a resource allocation problem in [LJ94]. Some basic theory about faults in the context of labelled transition systems can be found in a paper by Janowski [Jan94]. In [Jan97], the same author introduces fault-monotonic bisimulations in the context of CCS. We are not aware of any systematic approach to incremental verification, as we propose to develop in this project, although it is clear that many of the ideas put forward in the above cited papers are relevant. Also the large body of work on stepwise refinement [AL91, LV95, RE98] is clearly very relevant. A recent piece of work that appears to be particularly relevant is [KKLS00], which proposes a technique for incrementally constructing safety specifications, abstract algorithm descriptions, and simulation proofs.

As far as we know, the fault-tolerant systems literature mentioned above has not been applied to large examples and is not supported by tools. Nevertheless, several fault-tolerant protocols have been verified with some form of tool support, but usually without much underlying theory. For instance, the interactive theorem prover PVS has been used to verify several fault-tolerance

protocols in the domain of aircraft control, see [ORSH95] for a nice overview. Rushby [Rus97] presents a more systematic approach for a particular class of fault-tolerant protocols. First the algorithm is represented as a functional program and verified using PVS. Next it is transformed into an untimed synchronous system, and finally into a time triggered implementation. We refer to Section 6.2.2 for more references on tool support.

Our own background for this project is, among others, an assertional framework for the verification of distributed real-time systems [Hoo91, Hoo96a]. It has been applied to real-time protocols, first based on manual verification [Hoo93, Hoo94] and later supported by the interactive theorem prover PVS [Hoo95, Hoo96b, KHR97]. This real-time framework has also been extended to deal with fault-tolerant protocols, such as an atomic broadcast protocol [ZH95] and a membership algorithm for a dynamically changing network of processors [Hoo97].

The formal study of fault-tolerance started in an earlier, finished, STW/SION project “Fault Tolerance”. This has resulted in a compositional semantics for fault-tolerant real-time systems, and the verification of a simple triple modular redundancy example [CH92, CH93]. Moreover a trace-based compositional proof system for fault-tolerant systems has been devised [SH93, SH94].

Recently, concurrency control protocols have been proved to satisfy serializability. In [CHvdS99] a systematic way to extend these protocols with new actions and control information. We show that such an extension satisfies a few simple correctness conditions, the new protocol is serializable by construction.

Very related to the current project is recent work on the formal specification and mechanical verification of atomic commitment protocols (ACP’s) for distributed database systems. Timed state machines are used to specify the processes, whereas the communication mechanism between processes is defined using assertions. We verified a non-blocking ACP of Babaoglu and Toueg [BT93a] with our own recovery algorithm, since we found an error in the recovery mechanism of [BT93b]. The verification has been checked mechanically by means of PVS, but was in general rather ad hoc, investigating the feasibility of such complex verifications. It, however, also revealed the need for more underlying theory to be able to verify more complex applications more easily. Current work includes the verification of a new protocol combining concurrency control, centralized recovery (transactions may abort) and distributed recovery (volatile memory may get lost locally).

We also have much experience in the use of timed automata [AD94] and the I/O automata framework of Lynch et al [LT87, Lyn96] for protocol verification. Some representative case studies are [BPV94, DGRV00, GV98, HSV94, SV99a, SS00, Rom01]. Main themes in our research have been the further development of stepwise refinement methods, the extension of the I/O automata framework with real-time, hybrid and probabilistic features [LV96, SV99b, LSV01], and the further development of model checking technology for real-time systems [Feh99, BHV00, BFH⁺01b, BFH⁺01a, BFH⁺01c, HRSV01].

Finally, we expect to benefit from related projects in our group Informatics for Technical Applications (ITA), see <http://www.cs.kun.nl/ita/>, at the University of Nijmegen. Our group is well-known for its work on models and logics for specification and verification of state based systems. Unique about the group is that it brings together specialists on three different and important approaches in this area: I/O automata, Hoare’s logic, and coalgebras. The group has developed into a recognised center of excellence for the application of model checking and theorem proving technology, and has extensive experience with a large number of different verification tools such as Uppaal, Spin, SMV, Isabelle, and PVS. We participate in the European LTR project Verification of Hybrid Systems (VHS) and the European IST project VerifiCard. Relevant, for instance, is the intensive and successful collaboration in the context of VHS with the Universities of Aalborg and Uppsala, dealing with the extension and application of timed model checking technology. We also mention the new STW project HaaST on the verification of hard and softly timed systems in which methods and tools for timed and stochastic systems will be studied.

7 Work Programme

In general we interleave theoretical research and the application to case studies.

Year 1:

- (1 - 6) Detailed study of related theory about fault-tolerance, placing previously verified fault-tolerant protocols in a theoretical perspective.
- (7 - 12) Training in the use of verification tools. Treatment of a smaller case study of a fault-tolerant system.

Year 2:

- (13 - 18) Exploration of a large case study of a fault-tolerant system (probably from KPN Telecom) to get a good understanding of the system and the fault-tolerant protocols. Investigation of suitable tool support for visualization, simulation and model-checking of a simple version of the system.
- (19 - 24) Based on earlier theoretical investigations, already verified protocols and insight in the large case study, we propose a formal approach to the incremental verification of fault-tolerant real-time protocols.

Year 3:

- (25 - 30) Application of the approach to the case study, verifying a basic version of the protocols and gradually increasing the complexity of the failures that can be treated.
- (31 - 36) Evaluation of our approach based on the case study. Modifying the theory where needed and adapting the proposed tool support.

Year 4:

- (37 - 42) Applying the resulting framework to new examples of distributed real-time and fault-tolerant protocols.
- (43 - 48) Completion of PhD thesis.

8 a. Expected Use of Instrumentation

Experiments with powerful tools on complex protocols are part of this project. Some of these tools are very time and memory consuming. We need a powerful workstation or PC, especially for investigating how the amount of user interaction can be reduced.

9 Literature

Below five relevant papers from the applicants are listed.

- [CHvdS00] D. Chkhaev, J. Hooman, and P. van der Stok. Mechanical verification of a non-blocking atomic commitment protocol. In *Workshop on Distributed System Validation and Verification (DSVV'2000)*. Available on <http://www.cs.kun.nl/~hooman/ACP.html>, 2000.
- [GV98] W.O.D. Griffioen and F.W. Vaandrager. Normed simulations. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification*, Vancouver, BC, Canada, volume 1427 of *Lecture Notes in Computer Science*, pages 332–344. Springer-Verlag, June/July 1998. Full version available as [GV00].
- [Hoo97] J. Hooman. Verification of distributed real-time and fault-tolerant protocols. In *Algebraic Methodology and Software Technology (AMAST'97)*, pages 261–275. LNCS 1349, Springer-Verlag, 1997.

- [LV95] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.
- [ZH95] P. Zhou and J. Hooman. Formal specification and compositional verification of an atomic broadcast protocol. *Real-Time Systems*, 9(2):119–145, 1995.

10 Requested Budget

We ask for the standard budget for a PhD student, kf 228, and travel, kf 7.35. In addition, we ask $0.2 \times 4 \times \text{kf } 90 = \text{kf } 72$ to support the scientific programmer. As mentioned before (see Section 8), we need powerful equipment which goes above the normally available machine support (X-terminal or simple PC). Hence we require an additional 15 kf to be able to purchase a suitable workstation or powerful PC. The total budget thus amounts to kf 322.35.

References

- [AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
- [AH96] R. Alur and T.A. Henzinger, editors. *Proceedings of the 8th International Conference on Computer Aided Verification*, New Brunswick, NJ, USA, volume 1102 of *Lecture Notes in Computer Science*. Springer-Verlag, July/August 1996.
- [AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.
- [BBC⁺00] N. Bjørner, A. Browne, M.A. Colón, B. Finkbeiner, Z. Manna, H.B. Sipma, and T.E. Uribe. Verifying temporal properties of reactive systems: A STeP tutorial. *Formal Methods in System Design*, 2000. To appear.
- [BBM97] N. Bjørner, A. Browne, and Z. Manna. Automatic generation of invariants and intermediate assertions. *Theoretical Computer Science*, 173(1):49–87, 1997.
- [BFH⁺01a] G. Behrmann, A. Fehnker, T.S. Hune, K.G. Larsen, P. Pettersson, and J.M.T. Romijn. Efficient guiding towards cost-optimality in UPPAAL. In Margeria and Yi [MY01]. To appear.
- [BFH⁺01b] G. Behrmann, A. Fehnker, T.S. Hune, K.G. Larsen, P. Pettersson, J.M.T. Romijn, and F.W. Vaandrager. Minimum-cost reachability for priced timed automata. In Di Benedetto and Sangiovanni-Vincentelli [DBSV01]. To appear.
- [BFH⁺01c] Gerd Behrmann, Ansgar Fehnker, Thomas S. Hune, Kim G. Larsen, Paul Pettersson, and Judi Romijn. Guiding and cost-optimality in UPPAAL. In *The 2001 AAAI Spring Symposium Series: Model-Based Validation of Intelligence*. AAAI, 2001.
- [BGK⁺96] J. Bengtsson, W.O.D. Griffioen, K.J. Kristoffersen, K.G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. Verification of an audio protocol with bus collision using UPPAAL. In Alur and Henzinger [AH96], pages 244–256.
- [BHV00] G. Behrmann, T.S. Hune, and F.W. Vaandrager. Distributed timed model checking — how the search order matters. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 2000.
- [BLS96] S. Bensalem, Y. Lakhnech, and H. Saidi. Powerful techniques for the automatic generation of invariants. In Alur and Henzinger [AH96], pages 323–335.
- [BPV94] D.J.B. Bosscher, I. Polak, and F.W. Vaandrager. Verification of an audio control protocol. In Langmaack et al. [LRV94], pages 170–192.
- [BT93a] O. Babaoglu and S. Toueg. Non-blocking atomic commitment. In S. Mullender, editor, *Distributed Systems*, pages 147–168. Addison-Wesley Publishing Comp., 1993.
- [BT93b] O. Babaoglu and S. Toueg. Understanding non-blocking atomic commitment. Technical Report UBLCS-93-2, University of Bologna, 31 pages, <ftp://ftp.cs.unibo.it/pub/ozalp/Misc/NB-AtomicCommit.ps.gz>, 1993.
- [CASD95] F. Cristian, H. Aghili, R. Strong, and D. Dolev. Atomic broadcast: From simple message diffusion to Byzantine agreement. *Information and Computation*, 118:158–179, 1995.
- [CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Cambridge, Massachusetts, 1999.
- [CH92] J. Coenen and J. Hooman. A compositional semantics for fault-tolerant real-time systems. In *Proceedings Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 33–51. LNCS 571, Springer-Verlag, 1992.
- [CH93] J. Coenen and J. Hooman. Parameterized semantics for fault tolerant real-time systems. In J. Vytöpil, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 51–78. Kluwer Academic Publishers, 1993.
- [CHvdS99] D. Chkhaev, J. Hooman, and P. van der Stok. Serializability preserving extensions of concurrency control protocols. In *Proc. of the A. Ershov Third Int’l Conf. Perspectives of System Informatics*, pages 179–192. LNCS 1755, 1999.

- [CKdR92] Antonio Cau, Ruurd Kuiper, and Willem-Paul de Roever. Formalizing Dijkstra’s development strategy within Stark’s formalism. In Cliff B. Jones, Roger C. Shaw, and Tim Denvir, editors, *Proc. 5th. BCS-FACS Refinement Workshop*, 1992.
- [Coa90] B.A. Coan. Bibliography for fault-tolerant distributed computing. In B. Simons and A. Spector, editors, *Fault-tolerant Distributed Computing*, pages 274–298. LNCS 448, 1990.
- [Cri84] F. Cristian. Correct and robust programs. *IEEE Transactions on Software Engineering*, SE-10(2):163–174, 1984.
- [Cri85] F. Cristian. A rigorous approach to fault-tolerant programming. *IEEE Transactions on Software Engineering*, SE-11(1):23–31, 1985.
- [Cri89a] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3:146–156, 1989.
- [Cri89b] F. Cristian. Understanding fault-tolerant distributed systems. Research report rj 6980 (revised 10/16/90), IBM Almaden Research Center, 1989.
- [Cri91] F. Cristian. Reaching agreement on processor-group membership in synchronous distributed systems. *Distributed Computing*, 4:175–187, 1991.
- [Cri96] F. Cristian. On the semantics of group communication. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 1–21. LNCS 1135, Springer-Verlag, 1996.
- [DBSV01] M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors. *Proceedings Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC’01)*, Rome, Italy, Lecture Notes in Computer Science. Springer-Verlag, March 2001.
- [DGRV00] M.C.A. Devillers, W.O.D. Griffioen, J.M.T Romijn, and F.W. Vaandrager. Verification of a leader election protocol: Formal methods applied to IEEE 1394. *Formal Methods in System Design*, 16(3):307–320, June 2000.
- [Dij79] E.W. Dijkstra. A tutorial on the split binary semaphore. Technical Report EWD703 - 0, Burroughs, March 1979. Available via URL <http://www/cs.utexas.edu/users/EWD/index07xx.html>.
- [Feh99] Ansgar Fehnker. Scheduling a Steel Plant with Timed Automata. In *Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA’99)*. IEEE Computer Society Press, 1999.
- [GV98] W.O.D. Griffioen and F.W. Vaandrager. Normed simulations. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification*, Vancouver, BC, Canada, volume 1427 of *Lecture Notes in Computer Science*, pages 332–344. Springer-Verlag, June/July 1998. Full version available as [GV00].
- [GV00] W.O.D. Griffioen and F.W. Vaandrager. A theory of normed simulations. Technical Report CSI-R0013, Computing Science Institute, University of Nijmegen, July 2000.
- [HH95] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell HYbrid TECHnology Tool. In U.H. Engberg, K.G. Larsen, and A. Skou, editors, *Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, Aarhus, Denmark, volume NS-95-2 of *BRICS Notes Series*, pages 29–43. Department of Computer Science, University of Aarhus, May 1995.
- [Hol91] G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall International, 1991.
- [Hoo91] J. Hooman. *Specification and Compositional Verification of Real-Time Systems*. LNCS 558, Springer-Verlag, 1991.
- [Hoo93] J. Hooman. Specification and verification of a distributed real-time arbitration protocol. In *Proceedings 14th IEEE Real-Time Systems Symposium*, pages 284–293. IEEE, 1993.
- [Hoo94] J. Hooman. Compositional verification of a distributed real-time arbitration protocol. *Real-Time Systems*, 6(2):173–205, 1994.
- [Hoo95] J. Hooman. Verifying part of the ACCESS.bus protocol using PVS. In *Proceedings 15th Conference on the Foundations of Software Technology and Theoretical Computer Science*, pages 96–110. LNCS 1026, Springer-Verlag, 1995.
- [Hoo96a] J. Hooman. Assertional specification and verification. In M. Joseph, editor, *Real-time Systems: Specification, Verification and Analysis*, chapter 5, pages 97–146. Prentice Hall, 1996.

- [Hoo96b] J. Hooman. Using PVS for an assertional verification of the RPC-memory specification problem. In *Formal Systems Specification; The RPC-Memory Specification Case Study*, pages 275–304. LNCS 1169, Springer-Verlag, 1996.
- [Hoo97] J. Hooman. Verification of distributed real-time and fault-tolerant protocols. In *Algebraic Methodology and Software Technology (AMAST'97)*, pages 261–275. LNCS 1349, Springer-Verlag, 1997.
- [HRSV01] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. In Margeria and Yi [MY01]. To appear.
- [HSV94] L. Helmink, M.P.A. Sellink, and F.W. Vaandrager. Proof-checking a data link protocol. In H. Barendregt and T. Nipkow, editors, *Proceedings International Workshop TYPES'93*, Nijmegen, The Netherlands, May 1993, volume 806 of *Lecture Notes in Computer Science*, pages 127–165. Springer-Verlag, 1994.
- [Jan94] T. Janowski. Fault-tolerant bisimulation and process transformations. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 373–392. LNCS 863, 1994.
- [Jan97] T. Janowski. On bisimulation, fault-monotonicity and provable fault-tolerance. In *Algebraic Methodology and Software Technology (AMAST'97)*, pages 292–306. LNCS 1349, 1997.
- [JMS87] M. Joseph, A. Moitra, and N. Soundararajan. Proof rules for fault-tolerant distributed programs. *Science of Computer Programming*, 8:43–67, 1987.
- [JPXZ94] W. Janssen, M. Poel, Qiwen Xu, and J. Zwiers. Layering of real-time distributed processes. In Langmaack et al. [LRV94], pages 393–417.
- [KHR97] L. Kühne, J. Hooman, and W.P. de Roever. Towards mechanical verification of parts of the IEEE P1394 serial bus. In I. Lovrek, editor, *Proceedings of the 2nd International Workshop on Applied Formal Methods in System Design*, Zagreb, pages 73–85. University of Zagreb, Faculty of Electrical Engineering and Computing, 1997.
- [KKLS00] I. Keidar, R. Khazan, N. Lynch, and A. Shvartsman. An inheritance-based technique for building simulation proofs incrementally. In *Proceedings 22nd International Conference on Software Engineering (ICSE)*, Limerick, Ireland, June 2000. Full version submitted to a journal. Available via <http://theory.lcs.mit.edu/~roger/Research/research.html>.
- [Lam94] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.
- [Lam95] L. Lamport. TLA in pictures. *IEEE Transactions on Software Engineering*, 21(9):768–775, 1995.
- [Lam99] L. Lamport. Specifying concurrent systems with TLA+. In *Calculational System Design*, Amsterdam, 1999. IOS Press.
- [LJ92] Z. Liu and M. Joseph. Transformation of programs for fault-tolerance. *Formal Aspects of Computing*, 4(5):442–469, 1992.
- [LJ94] Z. Liu and M. Joseph. Stepwise development of fault-tolerant reactive systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 529–546. LNCS 863, 1994.
- [LMWF94] N. Lynch, M. Merritt, W. Weihl, and A. Fekete. *Atomic Transactions*. Morgan Kaufmann Publishers, Inc., 1994.
- [LRV94] H. Langmaack, W.-P. de Roever, and J. Vytupil, editors. *Proceedings of the Third International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, Lübeck, Germany, September 1994, volume 863 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.
- [LSV01] N.A. Lynch, R. Segala, and F.W. Vaandrager. Hybrid I/O automata revisited. In Di Benedetto and Sangiovanni-Vincentelli [DBSV01]. To appear.
- [LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.
- [LV95] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.

- [LV96] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, II: Timing-based systems. *Information and Computation*, 128(1):1–25, July 1996.
- [Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.
- [MBSU98] Z. Manna, A. Browne, H.B. Sipma, and T.E. Uribe. Visual abstraction for temporal verification. In A.M. Haeberer, editor, *Proceedings AMAST'98*, volume 1548 of *Lecture Notes in Computer Science*, pages 28–41. Springer-Verlag, 1998.
- [MBWB01] A. Mader, E. Brinksma, H. Wupper, and N. Bauer. Design of a PLC control program for a batch plant — VHS case study 1, 2001. Submitted to *European Journal of Control*. Available through URL <http://www.cs.kun.nl/~mader/papers.html>.
- [MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.
- [MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.
- [Mul93] S. Mullender, editor. *Distributed Systems*. Addison-Wesley Publishing Comp., 1993.
- [MY01] T. Margheria and W. Yi, editors. *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Genova, Italy, Lecture Notes in Computer Science. Springer-Verlag, 2001.
- [ORSH95] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.
- [RE98] W.P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge Tracts in Theoretical Computer Science 47. Cambridge University Press, 1998.
- [Rom01] J.M.T. Romijn. A timed verification of the IEEE 1394 leader election protocol. *Formal Methods in System Design*, 2001. Special issue on *FMICS'99*. To appear.
- [Rus97] J. Rushby. Systematic formal verification for fault-tolerant time-triggered algorithms. In C. Meadows and W. Sanders, editors, *Dependable Computing for Critical Applications 6*, pages 191–210, 1997.
- [SH93] H. Schepers and J. Hooman. Trace-based compositional reasoning about fault tolerant systems. In *Parallel Architectures and Languages Europe*, pages 197–208. LNCS 694, Springer-Verlag, 1993.
- [SH94] H. Schepers and J. Hooman. A trace-based compositional proof theory for fault tolerant distributed systems. *Theoretical Computer Science*, 128:127–157, 1994.
- [SS83] R. Schlichting and F. Schneider. Fail-stop processors: An approach to designing fault-tolerant computing systems. *ACM Transactions on Computing Systems*, 1(3):222–238, 1983.
- [SS00] D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. Technical Report CSI-R0009, Computing Science Institute, University of Nijmegen, May 2000. Conditionally accepted for *Springer International Journal on Software Tools for Technology Transfer (STTT)*.
- [SV99a] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems*, Bamberg, Germany, volume 1601 of *Lecture Notes in Computer Science*, pages 53–74. Springer-Verlag, 1999.
- [SV99b] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. Technical Report CSI-R9905, Computing Science Institute, University of Nijmegen, March 1999.
- [Tan81] A.S. Tanenbaum. *Computer networks*. Prentice-Hall International, Englewood Cliffs, 1981.
- [ZH95] P. Zhou and J. Hooman. Formal specification and compositional verification of an atomic broadcast protocol. *Real-Time Systems*, 9(2):119–145, 1995.