RWTH Aachen

Department of Computer Science
Technical Report

Revisiting Failure Detection and Consensus in Omission Failure Environments

Carole Delporte-Gallet and Hugues Fauconnier and Felix C. Freiling

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2005-13

RWTH Aachen · Department of Computer Science · June 2005

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

`http://aib.informatik.rwth-aachen.de/`

Revisiting Failure Detection and Consensus in Omission Failure Environments

Carole Delporte-Gallet[1], Hugues Fauconnier[1], and Felix C. Freiling[2]

[1] Laboratoire d'Informatique Algorithmique, Fondements et Applications, University Paris VII, France
[2] Laboratory for Dependable Distributed Systems, RWTH Aachen University, Germany

Abstract. It has recently been shown that fair exchange, a security problem in distributed systems, can be reduced to a fault tolerance problem, namely a special form of distributed consensus. The reduction uses the concept of security modules which reduce the type and nature of adversarial behavior to two standard fault-assumptions: message omission and process crash. In this paper, we investigate the feasibility of solving consensus in asynchronous systems in which crash and message omission faults may occur. Due to the impossibility result of consensus in such systems, following the lines of unreliable failure detectors of Chandra and Toueg, we add to the system a distributed device that gives information about the failure of other processes. Then we give an algorithm using this device to solve the consensus problem. Finally, we show how to implement such a device in an asynchronous system using some weak timing assumptions

1 Introduction

In systems with electronic business transactions, fair exchange is a fundamental problem. In fair exchange, the participating parties start with an item they want to trade for another item. They possess an executable description of the desired item and they know from which party to expect the desired item and which party is expecting their own item. An algorithm that solves fair exchange must ensure three properties: (1) every honest party eventually either delivers its desired item or aborts the exchange (*termination* property). (2) If no party misbehaves and all items match their descriptions then the exchange should succeed (*effectiveness* property). (3) If the desired item of any party does not match its description, then no party can obtain any (useful) information about any other item (*fairness* property). Fair exchange algorithms should guarantee these properties for mutually untrusted parties, i.e., even in the presence of arbitrary (malicious) misbehavior of a subset of participants. Therefore, fair exchange is usually considered a problem in the area of security.

It has recently been shown [4] that fair exchange, a security problem, can be reduced to a fault-tolerance problem, namely a special form of *consensus*. In the consensus problem, a set of processes must reach agreement on a single value out of a set of values, values which the individual processes have each proposed. The reduction from fair exchange to consensus holds in a model where each participating party is equipped with a tamper proof security module like a smart card. Roughly speaking, the security modules are certified pieces of hardware executing a well-known algorithm so they can establish confidential and authenticated channels between each other. However, since they can only communicate by exchanging messages through their (untrusted) host parties, messages may be intercepted or dropped. Overall, the security modules form a *trusted subsystem* within the overall (untrusted) system. The integrity and confidentiality of the algorithm running in the trusted subsystem is protected by the shield of tamper proof hardware. The integrity and confidentiality of data sent across the network is protected by standard cryptographic protocols. These mechanisms reduce the type and nature of adversarial behavior in the trusted subsystem to message loss and process self-destruction, two standard fault-assumptions known under the names of *omission* and *crash* in the area of fault-tolerance. To summarize, problems from the area of security motivate us to revisit the consensus problem in omission failure environments.

A central assumption for the reduction of fair exchange to consensus to hold is that the system be *synchronous*. A synchronous system has known upper bounds on all important timing parameters of the system like message delivery delay and relative process speeds. Synchronous systems are rare in practice. More common are asynchronous systems, i.e., systems with no or merely uncertain timing guarantees. This holds especially true for systems in which smart cards are used as security modules. Smart cards do not possess any device to reliably measure real-time since they are totally dependent on power supply from their host. If we would like to implement fair exchange using smart cards as security modules, we need an asynchronous consensus algorithm under the assumption of crash and omission faults.

In this paper, we investigate the feasibility of solving consensus in totally asynchronous systems in which crash and message omission faults may occur. Since a result by Fischer, Lynch, and Paterson [10] states that solving consensus deterministically is impossible even if only crash faults can happen, we must strengthen the model so that solutions are possible. We do this using the approach of unreliable failure detectors pioneered by Chandra and Toueg [6]. In this approach, the asynchronous model is augmented with a device that gives information about the failures of other processes. Failure detectors have proven to be a very powerful abstraction of timing assumptions that can express necessary and sufficient conditions for the solvability of problems in the presence of failures. In practice, we want to build a system that solves a certain problem (like consensus). So interesting for practical purposes is the question: What type of failure detector is sufficient to solve that problem? If such a failure detector is found, we only need to implement the failure detector to implement the algorithm in practice, usually reducing the complexity of solving the overall problem substantially. Interesting from a theoretical standpoint is the question: What type of failure detector is necessary to solve a problem? Answers to this question point to the minimum level of timing information which is needed to solve that problem. If only less is available, the problem is impossible to solve.

Here, we focus on the sufficiency part of the question, i.e., what type of failure detector is sufficient to solve consensus in asynchronous systems in which crash and omission faults can occur and what are the timing assumptions needed to solve Consensus. Omission faults, meaning that a process drops a message either while sending or while receiving it, were introduced by Hadzilacos [11] and later generalized by Perry and Toueg [14]. We make the following two contributions in this paper:

- We define a new type of failure detector, which we call Ω in analogy to [5], and give a protocol that solves consensus in omission failure environments as long as a majority of processes remains fault-free.
- We exhibit a set of weak timing assumptions in the spirit of earlier work [1, 3] that allow to implement Ω. More precisely, we show that the existence of some process with which every other process *eventually* can communicate in a timely way is sufficient to implement Ω.

The timing assumptions we exhibit are weaker than any other assumptions proposed up to now for the omission model. They therefore allow to implement consensus, and hence fair exchange, in a larger class of practical systems than before.

This paper is structured as follows: Section 2 introduces the system model, Section 3 specifies the new type of failure detector. Section 4 presents the algorithm to solve consensus using the failure detector from Section 3. Section 5 shows how to implement the failure detector under very weak synchrony assumptions. Finally, Section 6 concludes the paper.

2 Definitions and Model

We model a distributed system by a set of n processes $\Pi = \{p_1, p_2, \ldots, p_n\}$ that communicate using message passing over a network of channels in a fully connected topology. The communication primitives we assume are **send** and **receive**. Communication channels are reliable, i.e., every message sent is eventually received and every received message was previously sent. Processes can be faulty, as explained later.

We assume that the network is asynchronous, i.e., there is neither a bound on the relative process speeds nor on the message delivery delays. This means that while one process takes a single step within the execution of its local algorithm, any other process can take an arbitrary (but finite) number of steps. Also, messages can take an arbitrary (but finite) amount of time to travel from the source to the destination.

2.1 Failure Assumption

There are three ways in which processes can fail: (1) Processes can *crash*, i.e., they stop to execute steps of their local algorithm. Crashed processes never recover. (2) Processes can experience *send omission* failures, i.e., a message which is sent by a process is never placed into the communication channel. (3) Processes can experience *receive omission* failures, i.e., a message which arrives over the communication channel is never actually received by the algorithm of the process. Crash faults model, the usual hardware or operating system crashes, omission faults model overruns of internal I/O buffers within the operating system.

The types of failures result in three distinct failure assumptions:

- the *send omission model*, in which processes can crash and experience only send-omissions (and no receive omissions),
- the *receive omission model* (analogous to the send-omission model), and

– the *send/receive omission model* (sometimes also called *general omission*), in which processes can crash and experience either send-omissions or receive omissions.

A process p is *correct* if it does not make any failure at all, i.e., it is never crashed and experiences neither send nor receive omissions. Process p is *crash-correct* if it never crashes. If process p crashes at some time we say it is *crash-faulty*.

Due to the omissions, some processes could be disconnected forever from correct processes. More precisely, we say that process p is *in-connected*, if infinitely often it receives messages from some correct processes. In analogy, we say that process p is *out-connected*, if an infinity of its messages are received by some correct processes. A process is *connected* if it is in-connected and out-connected.

Clearly, in the send-omission failure model every process is in-connected, and in the receive omission failure model every process is out-connected.

2.2 Relations to Crash Model

Transient omissions refer to cases when a process regularly omits a message but equally regularly sends/receives a message over the channel. Such omissions can be masked by piggybacking information about previous messages on every new message sent over a channel.

```
Code for p:
on receive (m, d) from q
   if d = p ∧ m not delivered before then Receive m
   else if d ≠ p then send (m, d) to d

to Send(m) to d:
   send (m, d) to all
```

Fig. 1. Send/Receive with relay.

Since omissions introduce asymmetry in the communication relation, it is also an issue who can communicate with whom. For example, a process with receive omissions may receive messages from a correct process p but may fail to receive messages from another correct process q. We can mask parts of this asymmetry by using the relay algorithm of Figure 1 which defines new primitives **Send** and **Receive**. These primitives ensure that if a process p is in-connected then it receives infinitely often messages from all correct processes. Correspondingly, if a process is out-connected, then infinitely many of its messages are received by all correct processes. However note that the relay algorithm is costly concerning the communication load (each message from p to q generates $2n - 1$ messages).

In the following algorithms we avoid to use this relay algorithm. But it shows that if all crash-correct processes are connected, then by piggybacking old messages and with the relay algorithm all omissions can be masked and the omission models become equivalent to the crash failure model. Interesting cases arise if not all crash-correct processes are always connected.

2.3 Consensus

We use the standard definition of Uniform Consensus in this paper. The problem is defined using two primitives called *propose* and *decide*, both taking a binary value v. An algorithm solving consensus must satisfy the following properties:

- (Termination) Every correct process eventually decides.
- (Uniform Agreement) No two processes decide differently.
- (Validity) The decided value must have been proposed.

3 Failure Detectors for Omission Failure Environments

In this section we revisit failure detectors in crash environments and give a suitable definition for such a failure detector in omission failure environments.

The definition of failure detectors in the crash model are standard [6] and the literature contains a lot of definitions of failure detectors for crash failures. Among these, the failure detector Ω is particularly interesting: It has been proved to be the weakest failure detector to solve the consensus problem in the crash failure model with a majority of correct processes [5]. The output of Ω for each process p is the identity of one process, the assumed leader for p, such that eventually all correct processes have the same leader forever and this leader is a correct process. Hence Ω implements an *eventual leader election* .

We extend the definition of failure detector Ω to omission models, but some difficulties arise concerning the types of non faulty processes we considered. In the omission models, this definition is generally too restrictive because, it could be impossible to ensure that the chosen eventual leader does not experience permanent omissions. So we consider the following weaker definition:

Definition 1. *Failure detector Ω for omission models is a failure detector that outputs at each time for each process one process, called the leader, such that (1) there is a time after which, this leader is the same forever at all correct processes and (2) this process is crash-correct and connected.*

In the following algorithms the output of the failure detector Ω for process p is given by the value of local variable $Leader$.

4 Solving Consensus

We now show that the failure detector Ω introduced in the previous section is sufficient to solve consensus with a majority of correct processes in the send/receive omission model. Figure 2 depicts our consensus algorithm. It employs the well-known rotating coordinator paradigm, i.e., processes run through asynchronous rounds (counted using the variable r in task 1) and in every such round one process C is chosen as the coordinator. The processes start with v being their proposal value of consensus and spawn three concurrent tasks. In task 1, the coordinator is urged (by using $COORD$ messages) to "impose" its value on all processes by sending ONE messages (task 2). Processes then evaluate the value they receive from the coordinator (stored in $estfromC$). Unless it comes from the leader (referred to by Ω), a $\perp$ value is stored. In the second part of the algorithm, all processes broadcast their received value to all other processes (TWO

messages). If such messages are received from a majority of processes, the non-$\perp$ value given in the messages is the decided value and an appropriate decision message is broadcast to all. Task 3 just ensures that eventually all processes who receive the decision message actually do decide.

```
Code for p:
  Initialization:
    r := 0                                                    /* round number */
    v := ⟨proposed value⟩
  start Task 0 and Task 1 and Task 2
Task 0:
    upon receive(COORD, ∗, k) for the first time
        let (COORD, w, k) such a message
        send(ONE, w, k) to all

    upon receive(ONE, ∗, k) for the first time from another process
        let (ONE, w, k) such a message
        send(ONE, w, k) to all
Task 1:
  loop forever
    C := 1 + r mod n                                           /* coordinator */
    send(COORD, v, r) to p_C
    wait until (receive (ONE, ∗, r) from p_C) or (p_C ≠ Leader)
        if (ONE, w, r) is received then
          estfromC := w
        else
          estFromC := ⊥

    send(TWO, estFromC, r) to all
    wait until receive(TWO, ∗, r) from a majority of processes
        let L = {w | (TWO, w, r) is received }
        if L = {rec} for some rec ≠ ⊥ then
          send (DECIDE, rec) to all
          decide(rec)
          halt
        else
          if L = {rec, ⊥} for some rec ≠ ⊥ then
            v := rec
    r := r + 1

Task 2:
    upon received(DECIDE, k) from q
          send(DECIDE, k) to all
          decide(k)
          halt
```

Fig. 2. Consensus algorithm for the send/receive omission model using Ω.

Proposition 1. *If $Leader_p$ is the output of failure detector, algorithm of Figure 2 implements consensus for a majority of correct processes in the send/receive omission model.*

In the proofs of algorithms, by convention, given a variable x of process p, x_p^τ denotes the value of x in p at time τ.

To prove the proposition, we first state the two following lemmas:

Lemma 1. *If p and q end the first part (Lines 13 to 18) of a round r, then:*

(1) if $estFromC_p = x$ for some $x \neq \bot$ then $estFromC_q \in \{\bot, x\}$,

If p and q end Line 21 of a round r, then:

(2) if $L_p = \{x\}$ for some $x \neq \bot$ then $L_q = \{x\}$ or $L_q = \{x, \bot\}$,
(3) if $L_p = \{\bot, x\}$ for some $x \neq \bot$ then $L_q = \{x\}$ or $L_q = \{x, \bot\}$ or $L_q = \{\bot\}$.

Proof. (1): Notice first that for any process q, v_q is always a value proposed by some process and obviously $v_q \neq \bot$.

If $estFromC_p = x$ for some $x \neq \bot$ then p has received one message (ONE, x, r) from the coordinator $p_{1+r \bmod n}$. By the algorithm, the coordinator $p_{1+r \bmod n}$ sends only one message $(ONE, *, r)$ per round to all processes. Either the coordinator is not the leader for q ($p_{1+r \bmod n} \neq Leader_q$) and then $estFromC_q = \bot$, or the coordinator is the leader for q and q waits for the message ONE, and then $estFromC_q = x$.

(2) and (3): If $L_p = \{\bot, x\}$ or $L_p = \{x\}$ then at least one process, say u, ends the first part (Lines 13 to 18) of round r, and $EstFromC_u = x$. By (1), at most two values, $\bot$ and x, could be sent by processes to all processes in Line 19. And hence for any process q that ends round r either $L_q = \{x\}$ or $L_q = \{\bot, x\}$ or $L_q = \{\bot\}$. This concludes the proof of (3).

For (2), it remains to prove that if $L_p = \{x\}$ then $L_q \neq \{\bot\}$. As processes wait for a majority of processes, p and q get message from at least one common process s. By the algorithm s sends at most one message $(TWO, *, *)$ per round. Then s sends message (TWO, y, r) with either $y = \bot$ or $y = x$. As p and q have waited for this message, this excludes the case $L_p = \{x\}$ and $L_q = \{\bot\}$. ☐

Lemma 2. *If every process p begins some round r, with variable v equal to the same value d then all processes q ending this round either decides d or has $v_q = d$ at the end of this round.*

Proof. Consider such a round r. In this round, every message $COORD$ sent to the coordinator contains value d. Therefore, if the coordinator sends message ONE in round r, it sends (ONE, d, r). If a process p ends the first part of the algorithm (until Line 18), either it suspects the coordinator by Ω and then $estFromC_p = \bot$, or it receives message (ONE, d, r) from the coordinator and then $estFromC_p = d$. Hence, every message TWO sent in round r contains either d or $\bot$. Thus, for every p ending round r, either (a) $L_p = \{d\}$ and p decides d, or (b) $L_p = \{d, \bot\}$ and $v = d$ at the end of round r, or (c) $L_p = \{\bot\}$ and v does not change and remains equal to d. ☐

Now we show that the algorithm satisfies the properties of consensus.

Lemma 3. *The algorithm ensures the agreement property.*

Proof. Consider the first time a process, say p, sends a message $(DECIDE, d)$ for some d. By an easy induction, this sending occurs in Task 1, say in round r. In this round, after Line 21, L_p is $\{d\}$. Let q be any other process ending round r, by Lemma 1, in this round L_q is either $\{d\}$ and q decides in round r, or $\{d, \bot\}$ and q ends the round r with $v = d$.

By Lemma 2 and an easy induction, in every round $r' \geq r$, every process either decides d or ends the round with $v = d$. Hence, all processes which decide in Task 1, decide d. If a process decides in Task 2, by an easy induction, this decision is issued from a process which has decided in Task 1. This proves the agreement property. □

Lemma 4. *The algorithm ensures validity property.*

Proof. In the algorithm, all the processes send the values they have just received and by an easy induction they never insert in the algorithm a value of their own. □

Lemma 5. *The algorithm ensures termination.*

Proof. If there is no correct process, termination is trivial.

If any correct process decides by task 2 or task 1 then clearly all correct processes decide.

Assume that no correct process decides, then we prove that all correct processes participate to an unbounded number of rounds. For this, assume the contrary and let r_0 be the minimal round number in which at least one correct process is blocked forever. Let p be such a process in round r_0:

- p cannot be blocked in Line 14: if the current coordinator p_C is not crash-correct or is not connected, there is a time after which it cannot be leader and then p cannot be blocked. If the current coordinator is crash-correct and connected, by an easy induction p will eventually receive a ONE message from the coordinator.
- p cannot be blocked in Line 20: by an easy induction all correct processes will reach round r and send a TWO message for this round. As there is a majority of correct processes, p will receive a majority of TWO messages.

By the property of the eventual leader election, there is a time τ after which all correct processes have the same leader p_l and this leader is connected. Consider R the set of rounds in which correct processes are at this time τ. Let r_0 be the first round number such that p_l is the coordinator for r_0 and r_0 is greater than all elements of R. When all correct processes are in round r_0, they do not suspect coordinator p_l of the round r_0. Then they adopt for $estFromC$ the value sent by p_l. And so their L set is reduced to one element which is different from $\perp$ and they decide. □

This concludes the proof of the proposition.

5 Implementing Failure Detectors

In this section we give algorithms to implement eventual leader elections in the case of send and send/receive omissions. All these algorithms make some additional assumptions [6, 12, 13], that are needed if we want to implement consensus deterministically [10]. We also assume that all processes are able to measure time.[1]

[1] In fact they can measure time with a very low accuracy: it is sufficient that (1) the time interval measure is not decreasing (2) for each finite time interval I there is an integer n such that the measure for I is always less than n and (3) if the measure of interval time I is less than n then I is a finite time interval.

5.1 Partially Synchronous Models and Eventual Leader Election

In the omission models, messages from p to q are not received by q only due to send omissions from p or receive omission from q. Hence all communication links are assumed to be reliable. There is no duplication of messages and every message received has been sent before.

Concerning timeliness, a communication link (p, q) is *eventually timely* if there is a Δ and time τ_0 after which every message sent at time τ by p to q is received by time $\tau + \Delta$. Following [1, 3], we define eventual sources and bisources:

Definition 2. *Process p is an* eventual source *if and only if (1) p is a correct process and (2) for all correct processes q, communication link (p, q) is eventually timely.*

Process p is an eventual bisource *if and only if (1) p is a source and (2) for all correct processes q, communication link (q, p) is eventually timely.*

Note that if we have at least one eventual bisource in the system, the system is eventually rather synchronous: If all messages are broadcast and relayed one time, as eventually all links from correct processes to the eventual bisource and all the links from this eventual bisource to every correct process are eventually timely, there is a time after which all messages sent by correct processes are received in a timely way by all correct processes. Nevertheless, note that in the partially synchronous model of [9], it is assumed that eventually all links between processes are timely. This assumption is strictly stronger than the existence of an eventual bisource in the system. Having an eventual bisource does not exclude that the communication delay between two processes is unbounded if one of these process is faulty but crash-correct. For example, the communication delays from (faulty but crash-correct process) p to (correct process) q are unbounded, if p makes infinitely often send omissions to all processes but q, the communication from p to q (or every other processes to which q could relay messages from p) is not timely.

5.2 Eventual Leader Election

In the following, we assume for the send omission model that there is at least one eventual source and at least one eventual bisource for the send/receive or receive omission model. In these algorithms every process monitors the timeliness of the communication links. For this each process sends "ping" messages regularly and verifies that the messages arrive with a bounded delay. If this is not the case, the origin of the message is suspected to be faulty. But, even if all the ping messages from some process are received, due to the omission model, other messages from this process could not be received. Then in order to simplify the presentation we assume that all messages of the processes are piggybacked in the "ping" messages, in this way, if there is no omission of "ping" messages from p to q then there is no omission of any message from p to q.

Eventual Leader Election in the Send Omission Model. The algorithm in Figure 3 implements Ω for the case of send omission faults under the assumption that there is one eventual source.

In the algorithm, $Timer[q]$ is a special variable that is decremented at each clock tick. When $Timer[q]$ achieves a value equal to zero, we say that $Timer[q]$ expires. The principles of the algorithm are rather simple. Each process maintains a variable δ that is the assumed communication delay. This variable is incremented each time a communication of a process exceeds the assumed communication delay. Each process sends periodically (every η) a message to all others processes and maintains a vector V counting the number of times each process p exceeds the assumed communication delay δ. This vector is piggybacked in each message and each process updates its own vector V accordingly to the received vector (by taking the maximum of the two vectors). In this way, each vector V will evaluate the number of times a process exceeds the assumed communication delay. The leader will be the process having the minimal value in V (in case there is more than one such process, the process with the smallest identity is chosen).

Intuitively, if a process p makes an infinite number of send omission to some out-connected process, then eventually, the $V[p]$ of every out-connected crash-correct will be unbounded. However, if $V[p]$ is bounded by b for some out-connected crash-correct process, then it will be bounded by b for every out-connected crash-correct process. This proves that eventually all the $V[p]$ of out-connected crash-correct processes will be equal. Assuming that $V[p]$ is bounded for at least one process, choosing as leader the minimal p with the smallest value in vector V, ensures then that every out-connected crash-correct process eventually chooses p forever.

Then if s is an eventual source, it is straightforward to verify that $V[p]$ is bounded for every crash-correct process ensuring that every crash-correct process eventually chooses forever the same leader.

Note that this leader is not necessarily a correct process: if p makes infinitely often send omission to some process q that is not out-connected, it is possible that p is chosen as leader by all correct processes. In this case, the leader for q could be different from p.

If there is at least one eventual source in the system, this algorithm implement failure detector Ω:

Proposition 2. *In Algorithm of Figure 3, if there is at least one eventual source then there is a crash-correct out-connected l and a time after which every out-connected process has l as leader. Moreover, all correct processes receive infinitely often messages from l.*

We give here only a sketch of the proof:
By an easy induction we get:

Lemma 6. *If p is out-connected and q is crash-correct, then for all τ there exists $\tau' \geq \tau$ such that $V_p^\tau \leq V_q^{\tau'}$.*

Consider $\lim_{\tau\to\infty} V_p^\tau[q]$, as $V_p^\tau[q]$ is a non decreasing sequence of integers, either $\lim_{\tau\to\infty} V_p^\tau[q] = k$ for some integer k or $\lim_{\tau\to\infty} V_p^\tau[q] = \infty$. In the first case we say that $V[q]$ converges to k for process p, and in the second case that $V[q]$ does not converge for process p.

If p is crash-faulty or is out-disconnected, for every correct process, $Timer[p]$ will expire infinitely often and then $V[p]$ will be incremented infinitely often:

Initialization:

$\delta := 1$
for all q: V[q]:=0
for all $q : Timer[q] := \delta$

Task 1:

each η
  send V **to all**

Task 2:

on receive X **from** q
  for all $q : V[q] := \max\{V[q], X[q]\}$
  set $Timer[q]$ **to** δ

Task 3:

on $Timer[q]$ expired
  $V[q] := V[q] + 1$
  $\delta := \delta + 1$
  set $Timer[q]$ **to** δ

Task 4:

forever do
    $Leader := \min r$ such that $V[r] := \min\{V[q] | q \in \Pi\}$

Fig. 3. Implementation of Ω in a system with at least one eventual source and a majority of correct processes.

Lemma 7. *If p is crash-faulty or is out-disconnected then for all q crash-correct, $\lim_{\tau\to\infty} V_q^\tau[p] = \infty$.*

Lemma 8. *If $V[p]$ converges to k for some integer k and for some crash-correct out-connected q, then $V[p]$ converges to k for all crash-correct out-connected r.*

Let q out-connected crash-correct such that $\lim_{\tau\to\infty} V_q^\tau[p] = k$ and crash-correct process r such that $\lim_{\tau\to\infty} V_r^\tau[p] = \infty$ by Lemma 6, necessarily r is not out-connected, proving that $V[p]$ converges to k' for process r and $k \leq k'$. Conversely $k' \leq k$, proving the lemma.

Now consider an eventual source s, by definition there is a time τ_0 after which all messages sent by s arrive by some Δ, as for each time $Timer_q[s]$ expires, δ_q is incremented, there is a time $\tau_1 > \tau_0$ after which $\delta_q \geq \Delta$ or $Timer_q[s]$ never expires. Proving that $V_q[s]$ is bounded for all process q. By the previous Lemma, we get:

Lemma 9. *If s is an eventual source then $V[s]$ converges to k for some integer k and for all crash-correct processes out-connected.*

Hence, for at least one process q, $\lim_{\tau\to\infty} V_p^\tau[q] = k$ for all process p. By Lemma 7 and Lemma 8, let M be the max of all k such $V[r]$ converges to k for some r and p, there is a time τ_0 after which for all crash-correct out-connected p we have $V_p[r] = k$ if $V[r]$ converges to k and $V_p[r] > M$ if $V[r]$ does not converge. Then all crash-correct out-connected get the same leader forever. By Lemma 7, this leader is crash-correct and out-connected.

```
Initialization:
δ := 1
for all  q : Timer[q] := δ
for all  q, r : M[q, r] := 0
GoodInputs := ∅

Task 1:
each  η
   if (|GoodInput| ≤ n/2) then
      for all  q : M[q, p] := M[q, p] + 1
   send (M)  to all

Task 2:
on receive A  from  q
   for all x, y : M[x, y] := max{M[x, y], A[x, y]}
   add q to GoodInputs
   set Timer[q]  to  δ

Task 3:
on Timer[q] expired
   remove q  from  GoodInputs
   M[p, q] := M[p, q] + 1
   δ := δ + 1
   set Timer[q]  to  δ

Task 4:
forever do
   for all  r  do
      V[r] := min{max{M[q, r]|q ∈ L} such that |L| = ⌊n/2⌋ + 1}
   Leader := min r such that V[r] := min{V[q]|q ∈ Π}
```

Fig. 4. Implementation of Ω in a system with at least one eventual bisource and a majority of correct processes.

Eventual Leader Election for Send/Receive Omission Models For the algorithm of Figure 4, we assume that at least a majority of processes are correct and that there is at least one eventual bisource. The principles of this algorithm are similar to the previous one: each process approximates in δ a bound on communication delay. The main difference here is that processes maintain an array M to count the number of times messages from p to q exceeded the assumed bound. Moreover in order to ensure that the leader is in-connected it penalizes itself if it sees that it does not receive messages in a timely way from a majority of processes.

As processes may make receive omissions, the value of $M[p, q]$ does not necessarily mean that q has made $M[p, q]$ send omissions, then the choice of the leader is more intricate. For this, for each process q, we consider all the sets containing a majority of processes and for each such set the maximum value of the $M[p, q]$, then the estimate for q is the minimum of these values.

If there is at least one bisource in the system, this algorithm implements Ω:

Proposition 3. *In the Algorithm of Figure 4, if there is at least one eventual bisource there is a crash-correct connected l and a time after which every crash-correct connected process has l as leader.*

We again just give a sketch of the proof:

Note first that eventually information from out-connected processes reached all in-connected and crash-correct processes:

Lemma 10. *If p is out-connected and q is in-connected and crash-correct, then for all τ, there is τ' such that $M_p^\tau \leq M_q^{\tau'}$.*

If p is not in-connected and crash-correct, there is a time τ after which p does not receive any message from any correct process, as there is a majority of correct processes after time $\tau+\eta$ strictly less than $n/2$ processes belong to $GoodInputs_p$, and at each η, p increments for all q $M[q,p]$ and then $\lim_{\tau\to\infty} M_p[q,p] = \infty$ for all q. Then by Lemma 10:

Lemma 11. *If p is crash-correct and not in-connected then for all in-connected and crash-correct processes q and for all r $\lim_{\tau\to\infty} M_q^\tau[r,p] = \infty$*

If p is crash-faulty or not out-connected, there is a time after which no messages from p are received by correct processes and then for every correct process q $Timer[p]$ expires infinitely often, and $M_q[q,p]$ is incremented infinitely often and $\lim_{\tau\to\infty} M_q^\tau[q,p] = \infty$. By Lemma 10:

Lemma 12. *If p is crash-faulty or not out-connected then for all in-connected and crash-correct q:* $\lim_{\tau\to\infty} M_q^\tau[q,p] = \infty$.

As at least a majority of processes is correct, any subset of more than $n/2$ processes contains at least one correct process, then if p is crash-faulty or not out-connected or not in-connected by the previous lemmas, $\max\{M_q^\tau[r,p] | r \in L \text{ s.t. } |L| = \lfloor \frac{n}{2} \rfloor + 1\}$ is unbounded for every in-connected and crash-correct process q:

Lemma 13. *If p is crash-faulty or not out-connected or not in-connected then* $\lim_{\tau\to\infty} V_q^\tau[p] = \infty$ *for every in-connected and crash-correct process q.*

By lemma 10:

Lemma 14. *If* $\lim_{\tau\to\infty} V_q^\tau[p] = k$ *for some out-connected crash-correct q, then* $\lim_{\tau\to\infty} V_r^\tau[p] = k$ *for all in-connected crash-correct process r.*

Now let s be an eventual bisource, then there a Δ and a time τ after which, (1) every message sent by a correct process to s and (2) every message sent by s to any process correct p is received within Δ. Then as δ_s is incremented each time a timer expires, there is a time $\tau_s > \tau$ after which every correct process are in $GoodInputs_s$, as there is a majority of correct processes, after time τ_s $|GoodInputs_s| > n/2$ and s will not increment $M_s[p,s]$ for any p. In the same way, there is a time $\tau' > \tau_s$ after which no messages from s will exceed δ_p for any correct process p and then $M_p[p,s]$ will not increase. Then:

Lemma 15. *If s is an eventual bisource then for all in-connected crash-correct process p,* $\lim_{\tau\to\infty} V_p^\tau[s] < \infty$.

Hence, consider the set S of processes q such that for all correct p processes $\lim_{\tau\to\infty} V_p^\tau[q] < \infty$. From Lemma 13, S contains only crash-correct connected processes. By the previous lemma, if there is at least one bisource this set is not empty. By Lemma 14, for every $q \in S$ all the $\lim_{\tau\to\infty} V_p^\tau[q]$ for p correct are equal to, say k_q. Let q_0 be the process belonging to S with minimal identity such that k_q is minimal. It is easy to verify that eventually all correct processes will chose q_0 as leader. This concludes the proof.

6 Comparison with Previous Work and Conclusion

Failure detection and consensus in omission environments have been studied previously in unpublished work by Dolev, Friedman, Keidar and Malkhi [7, 8]. The failure detector $\Diamond\mathcal{S}(om)$ which they use to solve consensus is different but rather close in power to our definition of Ω. In contrast to [7, 8], we focus on the implementability of that failure detector under weak synchrony assumptions. To the best of our knowledge, our consensus algorithm using Ω is also novel.

Concerning timeliness assumptions enabling to solve consensus, Dwork, Lynch and Stockmeyer [9] proved that consensus is solvable if all correct processes are eventually timely. Other work [2] obtained the same timeliness assumptions as here. Note that in both cases, the authors consider the Byzantine failure model that is strictly stronger than omission faults. Also, these solutions do not use a modular approach with failure detectors.

In this paper we studied consensus in models where processes can crash and experience message omissions. This model was motivated from the area of security problems where omissions models can be used to model security problems with smart cards. In this paper we were mainly interested in proving the feasibility of solving consensus in such models, i.e., finding solutions, we were not interested in their efficiency. Hence, most of the algorithms presented here can probably be improved to ensure better performance. For example, in the case of send-omissions and implementation of Ω by algorithm of Figure 3, this algorithm could be improved: In Task 0, there is no need to relay of the messages ONE because with send-omissions the eventual chosen leader is not only in-connected but already receives infinitely many messages from correct processes.

One interesting open problem is to define the weakest failure detector to solve consensus with omission models, i.e., asking the rather fundamental question on what failure detector is necessary. In particular it is not proved that really the existence of an eventual bisource is needed for receive (and send/receive) omissions models.

The Ω implementation in the send omission model assumes only that there is at least one eventual source in the system, whereas for the receive or send-receive omission model we assume here that there is at least one eventual bisource. We conjecture that in the receive and send-receive omission models an eventual source is not enough.

References

1. M. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Stable leader election (extended abstract). In *Proceedings of the 15th International Symposium on Distributed Computing*, LNCS 2180, pages 108–122. Springer-Verlag, 2001.

2. M. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Consensus with byzantine failures and little system synchrony. Technical Report 2004-8, LIAFA, University Paris 7, 2004.
3. M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. On implementing 0mega with weak reliability and synchrony assumptions. In *22th ACM Symposium on Principles of Distributed Computing*, pages 306–314, 2003.
4. G. Avoine, F. C. Gärtner, R. Guerraoui, and M. Vukolic. Gracefully degrading fair exchange with security modules. In *In Proceedings of the 5th European Dependable Computing Conference(EDCC)*, Apr. 2005.
5. T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *J. ACM*, 43(4):685–722, July 1996.
6. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, Mar. 1996.
7. D. Dolev, R. Friedman, I. Keidar, and D. Malkhi. Failure detectors in omission failure environments. Technical Report TR96-1608, Cornell University, Computer Science Department, Sept. 1996.
8. D. Dolev, R. Friedmann, I. Keidar, and D. Malkhi. Failure detectors in omission failure environments (brief announcement). In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing (PODC97)*, 1997.
9. C. Dwork, N. A. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, Apr. 1988.
10. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, Apr. 1985.
11. V. Hadzilacos. *Issues of Fault Tolerance in Concurrent Computations*. PhD thesis, Harvard University, 1984. also published as Technical Report TR11-84.
12. A. Mostéfaoui, E. Mourgaya, and M. Raynal. Asynchronous implementation of failure detectors. In *DSN*, pages 351–360. IEEE Computer Society, 2003.
13. A. Mostéfaoui, S. Rajbaum, and M. Raynal. Conditions on input vectors for consensus solvability in asynchronous distributed systems. In *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing*, aug 2001.
14. K. J. Perry and S. Toueg. Distributed agreement in the presence of processor and communication faults. *IEEE Transactions on Software Engineering*, 12(3):477–482, Mar. 1986.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult `http://aib.informatik.rwth-aachen.de/` **or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email:** `biblio@informatik.rwth-aachen.de`

1987-01	*	Fachgruppe Informatik: Jahresbericht 1986
1987-02	*	David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes
1987-03	*	Manfred Nagl: A Software Development Environment based on Graph Technology
1987-04	*	Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment
1987-05	*	Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen
1987-06	*	Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*
1987-07	*	Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method
1987-08	*	Manfred Nagl: Set Theoretic Approaches to Graph Grammars
1987-09	*	Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents
1987-10	*	Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions
1987-11	*	Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory
1987-12		J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata
1988-01	*	Gabriele Esser, Johannes Rückert, Frank Wagner: Gesellschaftliche Aspekte der Informatik
1988-02	*	Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments
1988-03	*	Thomas Welzel: Simulation of a Multiple Token Ring Backbone
1988-04	*	Peter Martini: Performance Comparison for HSLAN Media Access Protocols
1988-05	*	Peter Martini: Performance Analysis of Multiple Token Rings
1988-06	*	Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze
1988-07	*	Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange
1988-08	*	Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks
1988-09	*	W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs
1988-10	*	Kai Jakobs: Towards User-Friendly Networking
1988-11	*	Kai Jakobs: The Directory - Evolution of a Standard
1988-12	*	Kai Jakobs: Directory Services in Distributed Systems - A Survey
1988-13	*	Martine Schümmer: RS-511, a Protocol for the Plant Floor

1988-14 * U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing

1988-15 * Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation

1988-16 * Fachgruppe Informatik: Jahresbericht 1987

1988-17 * Wolfgang Thomas: Automata on Infinite Objects

1988-18 * Michael Sonnenschein: On Petri Nets and Data Flow Graphs

1988-19 * Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers

1988-20 * Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen

1988-21 * Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment

1988-22 * Joost Engelfriet, Heiko Vogler: Modular Tree Transducers

1988-23 * Wolfgang Thomas: Automata and Quantifier Hierarchies

1988-24 * Uschi Heuter: Generalized Definite Tree Languages

1989-01 * Fachgruppe Informatik: Jahresbericht 1988

1989-02 * G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik

1989-03 * Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions

1989-04 * Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language

1989-05 J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)

1989-06 * Kai Jakobs: OSI - An Appropriate Basis for Group Communication?

1989-07 * Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems

1989-08 * Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control

1989-09 * Peter Martini: High Speed Local Area Networks - A Tutorial

1989-10 * P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation

1989-11 * Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science

1989-12 * Peter Martini: The DQDB Protocol - Is it Playing the Game?

1989-13 * Martine Schümmer: CNC/DNC Communication with MAP

1989-14 * Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements

1989-15 * M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments

1989-16 * G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling

1989-18 A. Maassen: Programming with Higher Order Functions

1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL

1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language

1990-01 * Fachgruppe Informatik: Jahresbericht 1989

1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)

1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas

1990-04 R. Loogen: Stack-based Implementation of Narrowing

1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies

1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication

1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work

1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation

1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke

1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine

1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit

1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains

1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)

1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars

1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars

1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit

1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen

1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations

1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences

1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming

1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990

1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence

1991-04 M. Portz: A new class of cryptosystems based on interconnection networks

1991-05 H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06 * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07 * Ludwig Staiger: Syntactic Congruences for w-languages
1991-09 * Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System
1991-10 K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming
1991-11 R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-12 * K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming
1991-13 * Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline
1991-14 * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15 J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16 J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17 A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-18 * Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems
1991-19 M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20 G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21 * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-22 H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL
1991-23 S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-24 R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems
1991-25 * Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks
1991-26 M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases
1991-27 J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28 J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30 T. Margaria: First-Order theories for the verification of complex FSMs
1991-31 B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-01 Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991
1992-02 * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen
1992-04 S. A. Smolka, B. Steffen: Priority as Extremal Probability
1992-05 * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories

1992-06 O. Burkart, B. Steffen: Model Checking for Context-Free Processes
1992-07 * Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems
1992-08 * Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line
1992-09 * Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme
1992-10 Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management
1992-11 Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines
1992-12 W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking
1992-13 * Matthias Jarke, Thomas Rose: Specification Management with CAD
1992-14 Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars
1992-15 A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)
1992-16 * Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik
1992-17 M. Jarke (ed.): ConceptBase V3.1 User Manual
1992-18 * Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems
1992-19-00 H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages
1992-19-01 G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)
1992-19-02 H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine
1992-19-03 C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus
1992-19-04 T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions
1992-19-05 S. Kahrs: Polymorphic Type Checking by Interpretation of Code
1992-19-06 M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine
1992-19-07 J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)
1992-19-08 D. Gärtner, A. Kimms, W. Kluge: pi-Red^+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus
1992-19-09 D. Howe, G. Burn: Experiments with strict STG code
1992-19-10 J. Glauert: Parallel Implementation of Functional Languages Using Small Processes
1992-19-11 M. Joy, T. Axford: A Parallel Graph Reduction Machine
1992-19-12 A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction

1992-19-13 K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)

1992-19-14 K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer

1992-19-15 S. Mintchev: Using Strictness Information in the STG-machine

1992-19-16 D. Rushall: An Attribute Grammar Evaluator in Haskell

1992-19-17 J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation

1992-19-18 W.S. Martins: Parallel Implementations of Functional Languages

1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)

1992-19-20 J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment

1992-19-21 S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)

1992-19-22 G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers

1992-19-23 S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)

1992-19-24 W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)

1992-19-25 M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting

1992-19-26 Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)

1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models

1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures

1992-19-29 G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)

1992-20 H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language

1992-21 H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language

1992-22 H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing

1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free

1992-24 K. Pohl: The Three Dimensions of Requirements Engineering

1992-25 * R. Stainov: A Dynamic Configuration Facility for Multimedia Communications

1992-26 * Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification

1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety

1992-28 * Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design

1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik

1992-30 A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic
1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems
1992-33 * B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: Euro-Bridge: Communication Services for Multimedia Applications
1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice
1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN
1992-36 M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis
1992-37 * K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models
1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES
1992-39 P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme
1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks
1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems
1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components
1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages
1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications
1993-01 * Fachgruppe Informatik: Jahresbericht 1992
1993-02 * Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems
1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments
1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES
1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis
1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik
1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions
1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases
1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking
1993-11 * R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels

1993-13 O. Maler, L. Staiger: On Syntactic Congruences for omega-languages

1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager

1993-15 M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept

1993-16 * M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain

1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes

1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing

1993-19 W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel

1993-20 * K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control

1993-21 M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle

1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993

1994-02 M. Lefering: Development of Incremental Integration Tools Using Formal Specifications

1994-03 * P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse

1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations

1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics

1994-06 * Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations

1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository

1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments

1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems

1994-11 A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems

1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars

1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems

1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition

1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents

1994-16 P. Klein: Designing Software with Modula-3

1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words

1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction

1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas

1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)

1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras

1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry

1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach

1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach

1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment

1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments

1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes

1995-01 * Fachgruppe Informatik: Jahresbericht 1994

1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES

1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction

1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries

1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types

1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases

1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies

1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures

1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages

1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency

1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases

1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology

1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views

1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work

1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems
1995-16 * W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming
1996-01 * Jahresbericht 1995
1996-02 Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees
1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability
1996-05 Klaus Pohl: Requirements Engineering: An Overview
1996-06 * M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs
1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming
1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents
1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines
1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation
1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell
1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems
1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming
1996-10 Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management
1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement
1996-12 * R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
1996-13 * K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools
1996-14 * R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996
1996-15 * H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases
1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization
1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet

1996-18 Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation

1996-19 * P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations

1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems

1996-21 * G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto

1996-22 * S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

1996-23 * M.Gebhardt, S.Jacobs: Conflict Management in Design

1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996

1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization

1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

1997-05 * S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting

1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets

1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

1997-11 * R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations

1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

1997-15 George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

1998-01 * Fachgruppe Informatik: Jahresbericht 1997

1998-02 Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes

1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

1998-04 * O. Kubitz: Mobile Robots in Dynamic Environments

1998-05 Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems

1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future
1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects
1998-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen
1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien
1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases
1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML
1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web
1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation
1999-01 * Jahresbericht 1998
1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version
1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager
1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing
1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference
1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie
1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL
1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures
2000-01 * Jahresbericht 1999
2000-02 Jens Vöge, Marcin Jurdzinski: A Discrete Strategy Improvement Algorithm for Solving Parity Games
2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach
2000-05 Mareike Schoop: Cooperative Document Management
2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling
2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages
2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 * Jahresbericht 2000
2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 * Jahresbericht 2001
2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09 Markus Mohnen: Interfaces with Default Implementations in Java
2002-10 Martin Leucker: Logics for Mazurkiewicz traces
2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 * Jahresbericht 2002
2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05	Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06	Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07	Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08	Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.